(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 December 2005 (08.12.2005)

PCT

(10) International Publication Number
WO 2005/116892 A1

(51) International Patent Classification[7]: G06F 17/60

(21) International Application Number:
PCT/US2005/014619

(22) International Filing Date: 27 April 2005 (27.04.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

10/853,306	24 May 2004 (24.05.2004)	US
10/852,926	24 May 2004 (24.05.2004)	US
10/853,546	24 May 2004 (24.05.2004)	US

(71) Applicant *(for all designated States except US)*: **APPLE COMPUTER, INC.** [US/US]; One Infinite Loop, M/S 3-PAT, Cupertino, CA 95014 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **GUIHENEUF, Bertrand** [FR/FR]; Apple, 68 rue du Fauboug St. Honoré, F-75008 Paris (FR). **MAURY, Sébastien** [FR/FR]; Apple, 68 Rue du Faubourg St. Honoré, F-75008 Paris (FR). **GUTKNECHT, Olivier** [FR/FR]; Apple, 68 Rue du Faubourg St. Honore, F-75008 Paris (FR). **JALON, Julien** [FR/FR]; Apple, 68 Rue du Faubourg St. Honoré, F-75008 Paris (FR). **RYDER, Scott** [US/US]; 1 Infinite Loop, M/S 82/EC, Cupertino, CA 95014 (US). **PATERSON, Toby** [GB/FR]; Apple, 68 Rue du Faubourg St. Honoré, F-75008 Paris (FR). **LEBEL, Jérôme** [FR/FR]; Apple, 68 Rue du Faubourg St. Honoré, F-75008 Paris (FR).

(74) Agents: **SCHELLER, James, C.** et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 12400 Wilshire Boulevard, 7th floor, Los Angeles, CA 90025 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHODS FOR SHARING GROUPS OF OBJECTS, SYNCHRONISING, AND SYNCHRONISING BETWEEN THREE OR MORE DEVICES

(57) Abstract: One aspect of the invention is a method of sharing a group of one or more objects between a plurality of users, in which one or more of said plurality of users is able to change parameter data of at least one said object. The method comprises storing at least one version of each said object; when an object is changed, creating a new version of the object, the new version of the object comprising additional data relating to the creation of the new version; storing the new version of the object together with any version of that object before the change; providing all versions of the object to each of said plurality of users; and using the additional data provided for each version of the object to determine how to display the object. Another aspect of the present invention comprises a method of synchronising data between a primary device and one or more subsidiary devices, said method comprising; storing a primary set of data on said primary device; comparing data on each subsidiary device with said primary set of data; updating said primary set of data; and updating data on each of said subsidiary devices using said updated primary set of data. Another aspect of the invention relates to a method of synchronising between three or more devices, said method comprising; storing an indication of the device or devices involved in each synchronisation event; storing data changes received during a current synchronisation event together with the device submitting those changes; and applying the data changes subsequent to the stored synchronisation event for the or each device.

WO 2005/116892 A1

METHODS FOR SHARING GROUPS OF OBJECTS, SYNCHRONISING, AND SYNCHRONISING BETWEEN THREE OR MORE DEVICES

RELATED APPLICATIONS

[0001] This application is related to and claims the benefit of United States Patent Application 10/853,546 entitled "METHOD FOR SHARING GROUPS OF OBJECTS," filed May 24, 2004. This application is related to and claims the benefit of United States Patent Application 10/852,926 entitled "A METHOD OF SYNCHRONISING," filed May 24, 2004. This application is related to and claims the benefit of United States Patent Application 10/853,306 entitled "A METHOD OF SYNCHRONISING BETWEEN THREE OR MORE DEVICES," filed May 24, 2004.

FIELD OF THE INVENTION

[0002] An aspect of the present invention relates to sharing a group of objects, such as a calendar of events or a collection of images, between two or more users. Preferably, each user is able to access the group by means of a networked device, such as a laptop or other computer.

[0003] Another aspect of the present invention relates to a method of synchronising. In particular but not exclusively, the present invention relates to synchronising data between devices such as computers, palm devices, personal digital assistants, music devices and mobile telephones. The data to be synchronised may comprise any data but commonly includes calendars, music files, photo files, emails, contact lists, bookmarks and any other such data. The present invention also encompasses synchronisation of applications. The present invention envisages that such synchronisation may occur between applications on the same device or on different devices. Hereinafter references to data includes data used by different applications and so the term devices includes applications stored on and run by an electronic device. Moreover, synchronisation between

devices includes synchronising data used by different applications on the same electronic device.

[0004] Another aspect of the present invention relates to a method of synchronising. In particular but not exclusively, the present invention relates to synchronising data between devices such as computers, palm devices, personal digital assistants, music devices and mobile telephones. The data to be synchronised may comprise any data but commonly includes calendars, music files, photo files, emails, contact lists, bookmarks and any other such data. The present invention also encompasses synchronisation of applications. The present invention envisages that such synchronisation may occur between applications on the same device or on different devices.

DESCRIPTION OF THE PRIOR ART

[0005] There are several available computer program applications that allow multiple users the facility of collaborative scheduling through sharing of calendars and the electronic sending of invitations. These include Lotus™, Exchange™, Meeting Maker™ and Outlook™. These known applications are designed, amongst other things, to allow users to view lists of attendees for meetings to which they are invited; to use e-mail to correspond with others; to send file attachments and respond to meeting and task requests; to schedule meetings and reserve locations and equipment; to share schedules with others in their workgroup; and to assign colleagues permission to read, modify or create items in their public or private folders.

[0006] As an example, in one known application a user may view the public calendars of all those he wishes to invite to a meeting in order to establish an appropriate time and place for the meeting. Based on this, the organiser may select a particular time and place for a meeting and invite various colleagues to the meeting. The invitees may accept or propose an alternative time and/or place. The organiser may then decide whether and how to reschedule the meeting and

transmit his decision to the invitees. Accordingly, all control for scheduling of the meeting is held by the organiser and the invitees act as little more than passive respondents to information.

[0007] In another known application, a server holds data for the calendars of a plurality of users. Each user may have one or more calendars. The holder of a calendar or an administrator assigns access to the calendar to the desired other users. The access may allow the other users to read the calendar only or to effect changes to, or edit, the calendar. To gain access, a user must be networked to the server.

[0008] In a further known application, a user stores the data for his calendar in his own user device, such as his laptop computer. Other devices could include a desktop computer, a Palm™ or other personal organiser and even a mobile phone. The user may elect to publish his calendar to a selected group of other users, for example using a .Mac™ server. To publish his calendar, the user must upload it to the server. Each time one of the group of other users logs on to the network and subscribes to the calendar, he is able to view the most recently published version of that calendar. He may then contact the first user to suggest changes to the calendar. However, he is not able to edit the calendar himself. Again, specific server software is required and all the calendar data must be stored in a predetermined format compatible with the server software and the communication protocol between the server and the devices on which the calendars are shared.

[0009] These well-known applications are heavily server-based and rely on particular calendar server software and communication protocols to allow users to share information on a network. A server will commonly store all information relating to all calendars in a particular format. Networked user devices must then access the server to obtain the required information. Any requests for changes to data must then be prepared in the correct format for successful transmission to the server. The server then effects the changes and stores the changed data for

subsequent forwarding to other devices on the network when requested. In particular, the server will overwrite existing data. More specifically, the calendar will typically be stored by the server as a single file, which is then overwritten when the calendar is changed.

[0010] This places a number of strictures on collaborative applications such as calendar sharing. In particular, all devices must be fully compatible with a particular server, which itself must be programmed with calendar server software. Moreover, to view a calendar or effect any change to it a user must be networked to the server. Accordingly, remote changes are not possible. Moreover, collaborative scheduling between two or more persons away from the office or other specific network environment is not feasible. Thus, known applications involve providing a closed environment in which a single copy of a server software application is used by different people.

[0011] Consequently, a problem arises when it is desired to allow users to edit one another's calendars and the user's devices are not networked to the same server. For example, it may be desired to allow a user to share his calendar with other users by means of a .Mac™ server, a RendezVous™ network and another generic server and to allow the other users to edit the calendar. The possible sources of change to one event in the user's calendar would include a change from a .Mac™ server, a change from a RendezVous™ server, a change from another generic server, a local change made by the user on his device, and a synchronisation operation occurring when a user synchronises the data on his devices. In addition, an invitation sent by another user may create an event. Clearly, in this situation the use of dedicated calendar server software is undesirable, if not unfeasible. Prior art calendar-sharing applications do not provide the functionality of sharing data using different networks.

[0012] Moreover, known dedicated calendar software and the concomitant communication protocols are unable to cope appropriately with changes made by users offline. For example, it would be possible for a first user to share his

calendar with a group of other users. It would be preferable to allow a second user to retrieve the shared calendar, edit the calendar offline and then share the change with the other users later, when he went online again. However, under these circumstances, before the second user went back on line, a third user could retrieve the shared calendar and make a different change to the calendar while still online. Despite being made after the second user's change and with the knowledge of more recent events, the third user's change would be shared before the second user's change and would be superseded by the second user's change when the second user went online again. This is undesirable and would lead to confusion.

[0013] It is particularly desirable to allow users to share calendars peer-to-peer. That is, to allow users to share calendars directly between devices without the intervention of a server. Prior art calendar-sharing applications do not provide this functionality.

[0014] Hitherto, synchronising such devices with another device has required the additional acquisition of computer software, usually developed by the manufacturer of one of the devices. Such computer software is relatively clumsy and inefficient. Moreover, the computer software usually places limitations on the type or format of the data to be synchronised. Finally, each device will have its own corresponding synchronisation software, each of which needs to be loaded on another device in order to effect synchronisation.

[0015] Synchronisation systems may be either servo based whereby the synchronisation system is stored and run on a server or central computer with the devices each synchronising to that server or computer. Alternatively, synchronisation can be achieved directly from one device to another and this is known in the art as "peer to peer" synchronisation.

[0016] With any synchronisation system, problems occurs when there are three or more devices. Such problems include when one of those devices is absent. Thus, when the absent device is brought for synchronisation, data for synchronisation

may be received by not only one other device but also all of the other devices in the system. This leads inherently to excessively redundant synchronisation procedures and increases the likelihood of errors being introduced. One solution for obviating the problem is disclosed in one of our co-pending US patent applications holding serial no. 10/453,051 filed on 2nd June 2003.

[0017] As well as the various different types of data to be synchronised, it is also possible to consider synchronising not only each record but instead particular fields of a record. One advantage of synchronising only the fields is that there is a smaller data exchange involved in the synchronisation process. Another advantage is that two devices may change a different field in the same record without any conflict occurring. If synchronisation was effected on a record basis, then in this situation a conflict would occur.

[0018] In addition, not only can the attributes be synchronised between devices but also the relationship between those attributes. For example, consider contact lists where a person's contact details are given and include home telephone number, work telephone number and mobile telephone number as well as various addresses including Email addresses of both work and home and postal address and work address. Each of the contact details would be considered a field whereas all of the contact details for a particular person would be considered the record. The contact lists may also include the relationships between that person and other persons held in the contact lists. This could include the fact that the first person is a brother to a second person. A third person's details may also be given together with the relationship that he is a father to both the first and second person. Any type of relationships may be given, not just relative relationships but also relationship information such as girlfriend, boyfriend or partner, work, colleague or other contact relationship.

[0019] One type of known synchronisation system is shown in Figure 13. In Figure 13, there is a first device 2002 to be synchronised with a second device 2004. The devices 2002 and 2004 are due for synchronisation through a

synchronisation engine 2006. The synchronisation engine uses conduits 2008 and 2010, a synchroniser 2012 and a mingler 2016. Each of the conduits has a device specific area 2008a, 2010a and a structured delta area 2008b and 2010b. Each of the conduits has a conduit store 2014, only one of which is shown.

[0020] The synchronisation system can be separated into three parts: the synchronisation software which is stored on the devices, the synchronisation engine which includes the synchroniser and mingler, and the conduits. The synchronisation software provides the usual user interface for receiving and prompting for instructions from a user. The user interface enables synchronisation to be initiated, provide a format for resolving conflicts, registering and configuring the devices to be included in the synchronisation system and the synchronisation log.

[0021] The synchroniser effects the synchronisation of the data by processing the changes. Preferably, the data comprises a field but a whole record may be used if desired.

[0022] The conduits act a liaison between the synchroniser and the devices. The conduits principally translate the data between the devices data format and the synchronisers canonical format. That is to say, the conduit receives data to be synchronised from the respective device and puts it into a canonical format and submits the same to the synchroniser. Conversely, the conduits receive canonical formatted data which is to be used to update the device and converts the same into the format of the respective device. In the example of contact lists, the device format may include fields such as first name, last name etc., whereas the canonical format for the synchroniser comprises fn for first name and ln for last name.

[0023] The conduit provides a static description of the device's capabilities and provides that to the synchroniser. The description does not change dynamically over time. Thus, it can provide the synchroniser with what type of records or

fields the device can synchronise and the list of fields for each record type supported by the device.

[0024] The structured delta 2008b, 2010b of each conduit retrieves the record or field which has been modified in the device and compares it with that stored in the store 2014. The structured delta effects that comparison and passes the change in the form known as a delta Δ. Thus, each of the conduits 2008 and 2010 provide a stream of deltas to the synchroniser.

[0025] In some devices, most typically mobile telephones, the devices are arranged to conserve memory as much as possible. Thus, many fields are truncated. Hence, there are seeming differences between that stored in the conduit store 2014 and that stored on the device. An example of such a truncation would be to only allocate a certain number of letters in the person's name in a contact list. For example, the name Gardio Freedman (which is stored in the conduit store 2014) is truncated by the device 2004 to Gardio Freed. Thus, another function of the conduit is to include in the description of the device the type of truncation or translation of any data which may occur by the device. Thus, when receiving data from the device, the conduit should emulate the device and store the truncated data. That truncated data together with the description of the truncation or translation rules enables to conduit to prepare the full data for comparison to correctly identify true deltas Δs.

[0026] Thus, when synchronisation is initiated, the conduit receives data to be synchronised from the respective device and translates any records which have been truncated. The structured delta then retrieves the stored record from the conduit store 2014 and compares that with that received and translated from the device and prepares the change in the form of a delta Δ. The stream of deltas is of course presented in the canonical format prior to submission to the synchroniser. The synchroniser passes those streams of deltas to other devices. In return, the conduit also receives deltas from other devices, translates them

from canonical format to the devices' format including any truncation to be applied and updates the device.

[0027] There are two types of synchronisation, fast synchronisation and slow synchronisation. In fast synchronisation, the conduit provides merely the changes in fields or records since the last synchronisation. Those changes may include any fields or records which have been added, modified or deleted. This is the default-type of synchronisation and the one that is preferred since it involves less data transfer and is significantly quicker. However, not all devices can support this type of synchronisation. The second type of synchronisation is referred to as slow synchronisation. In slow synchronisation, the conduit is unable to identify which fields or records have been changed since the last synchronisation. Accordingly, all data in the device is passed for synchronisation. The synchronisation engine must identify those changes by comparing each and every record with that stored in the conduit store 2014. Needless to say, slow synchronisation is relatively slow and inefficient in comparison to fast synchronisation.

[0028] As noted previously, any changes to the records or the fields may involve a deletion, addition or modification. Figure 14 is representative of two devices, device 1 and device 2, submitting such changes. In this case, D indicates a deleted recorded, M indicates a modified record and A indicates a record to be added. The tables include the record ID to be deleted, modified or added. Thus, device 1 in the example shown in Figure 14 deletes record 7, modifies record 5 and adds record 1, and these changes are submitted to the synchronisation engine 2006. A second device submits other changes to the synchronisation engine 2006 including modification of record 9, modification of record 3 and adding record 2. These changes are submitted either during a fast synchronisation procedure or a slow synchronisation procedure. In either case, the synchronisation engine needs to apply the corresponding changes submitted by device 2 to device 1. Thus, the synchronisation engine 2006 supplies to device 1 the changes submitted by

device 2. Namely, modification of record 9, modification of record 3 and addition of record 2. Corresponding changes are passed to device 2, which have been submitted by device 1. Each device through its conduit 2008, 2010 goes through all records supported by the device through the record IDs from the first record ID to the last.

[0029] Several problems occur with this existing arrangement for synchronisation. The first such problem is when a device is absent or application not available from the synchronisation event. In this case, any unavailable devices are assumed to be present and a virtual output is generated by the synchroniser. This virtual output is stored in a virtual store 2018 in the synchronisation engine. When the absent device is available to the synchronisation system, the virtual output stored in the virtual store 2018 is used as input to the synchronisation engine 2006 to update the absent device, device 3. This leads to a further second problem in that if both device 1 and device 2 both effect the same change to the same field or record, then potentially redundant synchronisation steps are required when updating absent device 3.

[0030] In all of the above, should any change submitted by more than one device be in conflict with each other, then those changes are submitted for conflict resolution through the user interface.

[0031] One known solution for ameliorating the problem of an absent device and then subsequent redundant synchronisation steps is to effect likelihood matching between records instead of comparing all of the records in the device to be updated. However, this is relatively inefficient and does not obviate all of the potentially redundant synchronisation steps.

[0032] A third problem associated with prior synchronisation systems is as a consequence of the truncation of data by the devices. In the former example, such truncation includes restricting the number of letters in a persons name in a contact list, i.e. the name Gardio Freedman is stored as Gardio Freed. Hitherto, this problem has been overcome by first comparing the fields between the device

and its respective conduit store and if there is a match between the two fields, then any truncation of data is to be ignored. If the two fields are not comparable, then the devices specific areas 2008a and 2010a get the full record from the device and compare that with that stored in the conduit store.

[0033] However, this apparent solution further compounds the problem of when one of the devices is absent, since when the absent device is brought for synchronisation the problem of device truncation is repeated. Moreover, this solution is very data intensive and hence very time consuming.

[0034] A further fourth problem associated with existing synchronisation systems is that synchronisation of relationship data is very limited. This is as a consequence of the limitations imposed by the conduits.

[0035] A further fifth problem associated with existing synchronisation systems results from the fact that the synchronisation software and conduits all reside within the synchronisation engine 2006. Hence, if there is any problem with the synchronisation software, then no devices can be synchronised. Moreover, when two or more devices are connected to the synchronisation engine and undergoing synchronisation, the devices are synchronised simultaneously and hence the same data may be accessed at the same time leading to greater conflicts and greater error generation.

[0036] Hitherto, synchronisation of a number of devices has involved a large number of comparison steps between actual data or changes in data. Such comparison has potentially involved extensive redundant comparison steps. Another problem with the existing synchronisation methods is that such synchronisation has been effected based on the systems clocks defined by the devices. Those systems clocks can be modified either intentionally or malevolently with disastrous consequences for subsequent synchronisation of the devices.

[0037] Synchronisation between devices may be effected in a number of ways. Once such system is disclosed in one of our co-pending US patent applications

holding Serial No. 10/453051 filed on 2nd June 2003. A further system is described in one of our other co-pending US patent applications filed contemporaneously and the contents of which are incorporated herein by reference. However, to assist in the appreciation of a synchronisation method, a brief description will now be given with reference to Figure 18. In Figure 18, there is shown a first device 3002 to be synchronised with a second device 3004. The devices 3002 and 3004 are due for synchronisation through a synchronisation engine 3006. The devices communicate with the synchronisation engine 3006 through conduits 3008 and 3010. Each of the conduits have a device specific area 3008a, 3010a, and a structured delta area 3008b and 3010b. Typically, the conduits also include a conduit store 3014, only one of which is shown in connection with conduit 3010.

[0038] The synchronisation engine 3006 includes a synchroniser 3012, mingler 3016, Truth Table 3020 and schema 3022.

[0039] The conduits act as a liaison between the synchronisation engine and the respective device. The conduits principally translate data between the devices data format and the synchronisation engine's canonical format. That is to say, the conduit receives data to be synchronised from the respective device and puts it into a canonical format and submits the same to the synchronisation engine 3006. Conversely, the conduits receive canonical formatted data which is to be used to update the device and converts the same into the format of the respective device.

[0040] The device specific areas 3008a, 3010a, of each of conduits contain a static description of the devices' capabilities and indicates what types of records or fields can be synchronised and the list of fields for each record type which is supported by the respective device.

[0041] The structured deltas 3008b, 3010b retrieve the record or field which has been modified in the device and compares it with that stored in the conduit store 3014. The structured delta effects the comparison and passes the change in the form known as a delta to the synchronisation engine 3006.

[0042] The mingler 3016 receives the stream of deltas from each device in turn through the respective conduit and updates the Truth Table 3020. The Truth Table is an amalgamated copy of the records from all of the devices involved in the synchronisation system. Thus, during synchronisation, each device is synchronised serially one at the time with the Truth Table and each record of the device being synchronised is synchronised with each record of the Truth Table. Hence, having obtained an amalgamation of all of the updated records from all of the devices, only then are the devices synchronised with the Truth Table, again serially.

[0043] Should any conflicts occur, then such conflicts are passed for conflict resolution through a user interface (not shown). The synchronisation engine 3006 also includes a synchroniser for effecting the functions of the mingler 3016 and its affect on the Truth Table 3020. In addition, the synchroniser 3012 manages the communications to and from the conduits 3008, 3010. Finally, the schema 3022 enables a user of the synchronisation system to define the data to be synchronised, again through the user interface.

[0044] As discussed above, synchronisation of data involves a large number of comparison steps. The excessively large number of comparison steps can be exacerbated when any device involved in the synchronisation system is absent when the others are undergoing synchronisation.

[0045] Hitherto, synchronisation methods have been effected based upon the systems clocks defined by the devices. As noted earlier, those systems clocks can be modified either intentionally or malevolently. Thus, some devices, whose systems clocks have been changed, may be synchronised needlessly or, indeed, some data may not be included in the synchronisation.

SUMMARY OF THE INVENTION

[0046] Aspects of the present invention is made with a view to overcoming these difficulties. In particular, an aspect of the present invention provides a method of

sharing a group of one or more objects between a plurality of users, in which one or more of said plurality of users is able to change parameter data of at least one said object. The method comprises storing at least one version of each said object; when an object is changed, creating a new version of the object, the new version of the object comprising additional data relating to the creation of the new version; storing the new version of the object together with any version of that object before the change; providing all versions of the object to each of said plurality of users; and using the additional data provided for each version of the object to determine how to display the object. The group may be a calendar and each object may be an event in the calendar. In that case, the object parameter data may comprise a start time of the event, an end time of the event, a description of the event, a status of the event, whether the event is to be repeated and the persons attending the event. The additional data may comprise an identification of the user who made the change, a time at which the change was made, a description of the change, a user comment relating to the change and an identification of the previous version of the event from which the present version was created.

[0047] Aspects of the present invention relate to a method of synchronising. In particular but not exclusively, the present invention relates to synchronising data between devices such as computers, palm devices, personal digital assistants, music devices and mobile telephones. The data to be synchronised may comprise any data but commonly includes calendars, music files, photo files, emails, contact lists, bookmarks and any other such data. The present invention also encompasses synchronisation of applications. The present invention envisages that such synchronisation may occur between applications on the same device or on different devices.

[0048] An aspect of the present invention is particularly directed to overcoming five problems and these include accommodating for absent devices or applications, avoiding redundant synchronisation steps, accommodating for

truncation or translation of data by devices or applications, enabling and preserving synchronisation of relationships and obviating known synchronising methods wherein two devices or applications are accessing the same database thereby causing inherent conflict.

[0049] An aspect of the present invention is directed to providing an improved method of synchronising which overcomes or ameliorates each of the problems enumerated above. That said, the present invention comprises a method of synchronising data between a primary device and one or more subsidiary devices, said method comprising: storing a primary set of data on said primary device; comparing data on each subsidiary device with said primary set of data; updating said primary set of data; and updating data on each of said subsidiary devices using said updated primary set of data.

[0050] Hitherto synchronisation of a number of devices has involved a large number of comparison steps between actual data or changes in data. Such comparison has potentially involved extensive redundant comparison steps. Another problem with existing synchronisation methods is that such synchronisation has been effected based upon the systems clocks defined by the devices. Those system clocks can be modified either intentionally or malevolently with disastrous consequences for subsequent synchronisation of the devices.

[0051] Thus, another aspect of the present invention seeks to improve methods of synchronising by optimally identifying only those comparison steps which need to be made. Accordingly, the present invention relates to a method of synchronising between three or more devices, said method comprising:storing an indication of the device or devices involved in each synchronisation event; storing data changes received during a current synchronisation event together with the device submitting those changes; and applying the data changes subsequent to the stored synchronisation event for the or each device.

BRIEF DESCRIPTION OF THE DRAWINGS

[0052] Embodiments of the present invention will now be described by way of further example only and with reference to the accompanying drawings, in which:

[0053] Figure 1 is a view of a calendar user interface to which the present invention may be applied;

[0054] Figure 2 is a schematic view of a network for sharing a calendar according to the present invention;

[0055] Figure 3A is an illustration of a series of changes to an event according to the prior art;

[0056] Figure 3B is an illustration of a series of changes to an event according to the present invention;

[0057] Figure 4 is an illustration of an object according to the present invention;

[0058] Figures 5A to C are illustrations of the synchronisation of an event between different calendar-sharing devices;

[0059] Figure 6 is a schematic view of another network for sharing a calendar according to the present invention;

[0060] Figure 7 is a table showing the transfer of information between three computers according to the present invention;

[0061] Figure 8 is another table showing the transfer of information between three computers according to the present invention;

[0062] Figure 9 is a view of an inspector box and a history box in a calendar user interface according to the present invention;

[0063] Figures 10A and B show changes to a history box in a calendar user interface according to the present invention;

[0064] Figure 11 is a view of a notification box in a calendar user interface according to the present invention; and

[0065] Figure 12 is another table showing the transfer of information between three computers according to the present invention.

[0066] Figure 13 is a schematic overview of a synchronisation system according to the prior art;

[0067] Figure 14 is a diagram illustrating changes effected on three devices being synchronised according to the prior art;

[0068] Figure 15 is a schematic diagram of a synchronisation system according to the present invention;

[0069] Figure 16 is a figure illustrating changes effected by three devices being synchronised according to the present invention;

[0070] Figure 17 illustrates a schematic diagram of a system overview of synchronisation according to the present invention.

[0071] Figure 18 is a schematic diagram of a synchronisation system as disclosed in our co-pending application filed contemporaneously;

[0072] Figure 19 is a schematic diagram of an Administration Table and Truth Table for effecting the method of synchronising data according to the present invention;

[0073] Figure 20 is a schematic diagram of an Administration Table and Truth Table for effecting the method of synchronising data according to the present invention;

[0074] Figure 21 is a schematic diagram of an Administration Table and Truth Table for effecting the method of synchronising data according to the present invention;

[0075] Figure 22 is a schematic diagram of an Administration Table and Truth Table for effecting the method of synchronising data according to the present invention;

[0076] Figure 23 is a schematic diagram of an Administration Table and Truth Table for effecting the method of synchronising data according to the present invention;

[0077] Figure 24 is a schematic diagram of an Administration Table and Truth Table for effecting the method of synchronising data according to the present invention;

[0078] Figure 25 is a schematic diagram of an Administration Table and Truth Table for effecting the method of synchronising data according to the present invention;

[0079] Figure 26 is a schematic diagram of an Administration Table and Truth Table for effecting the method of synchronising data according to the present invention;

[0080] Figure 27 is a schematic diagram of an Administration Table and Truth Table for effecting the method of synchronising data according to the present invention;

[0081] Figure 28 is a schematic diagram of an Administration Table and Truth Table for effecting the method of synchronising data according to the present invention;

[0082] Figure 29 is a schematic diagram of an Administration Table and Truth Table for effecting the method of synchronising data according to the present invention;

[0083] Figure 30 is a schematic diagram of an Administration Table and Truth Table for effecting the method of synchronising data according to the present invention; and

[0084] Figure 31 is a schematic diagram of a Truth Attribute Table and Truth Relationship Table to illustrate in more detail the Truth Table included in the method of synchronising data according to the present invention.

DETAILED DESCRIPTION

[0085] Figure 1 is an illustration of a calendar user interface (UI) 10 to which the present invention may be applied. The UI 10 comprises a calendar main view 20, which displays events 75 over a selected time frame for one or more calendars.

As shown by the calendar list 60, in the present example three calendars are displayed in the main view 10. These calendars are the user's work calendar 71, DVD release calendar 72 and private calendar 73. The private calendar 73 is shown in bold in the calendar list 60 and, correspondingly, events in the private calendar 73 are shown in bold in the calendar main view 20. The UI 10 further comprises an inspector window 30, which allows the parameters of a particular event in a calendar to be viewed. In the present example, the inspector 30 shows the scheduled time of a dentist appointment. Finally, the UI 10 includes a history window 40 and a notification window 50. These will be discussed in more detail below.

[0086] In a preferred embodiment of the present invention, each calendar may be shared by a plurality of users using one or more networks. Figure 2 illustrates exemplary networks for sharing a calendar according to the present invention. In effect, Figure 2 shows three interlinked networks 130, 135 and 140. The first network 130 comprises a laptop computer 101 and a desktop computer 105 networked over the Internet by means of a .Mac™ server 131. The second network comprises the laptop computer 101 networked to a second desktop computer 110 in a RendezVous™ network. A RendezVous™ network is a wireless network that eliminates the need to enter complicated Internet Protocol (IP) addresses or other information to set up devices on a network or locate people and other resources. Thus, when a "RendezVous™" device enters a RendezVous™ network, it is automatically detected and included in the network. Communications between devices in a RendezVous™ network are effected by wireless transmissions 136. The third network 140 comprises the second desktop computer 110, a local area network (LAN) server 141 and two further desktop computers 115 and 120 connected to the LAN server 141.

[0087] Each of the three user computers 101, 105 and 110 in the first and second networks may have a different user. For example, laptop 101 may have a first user, desktop 105 may have a second user and laptop 110 may have a third user.

In this example, the second user is the wife of the first user and the third user is a work colleague of the first user. The first user has the three calendars shown in Figure 1. As indicated by the "people" icons in the calendar list 60, the user shares his work and personal calendars with both the second and third users. That is, the second and third users are able to view the work and personal calendars of the first user at desktop computer 105 and desktop computer 110 respectively. Moreover, they are able to edit the work and personal calendars at their respective desktop computers 105 and 110 and to share the edited calendar with the other users.

[0088] For example, if the dentist were to contact the second user and change the time of the appointment to 12:00 PM, the second user would be able to view the first user's private calendar and edit the dentist appointment. Assuming both the first and second users are online, the change would then automatically be transmitted from the first desktop 105 to the laptop 101 by means of the .Mac™ server 131 and from the laptop 101 to the second desktop 110 by means of the RendezVous™ network. Accordingly, each user is able to view and edit any shared calendar and all changes are automatically transmitted to the other users.

[0089] In addition, the third user is also able to make changes to the first user's calendars and the changes are then shared with first user over the RendezVous™ network and subsequently with the second user by means of the .Mac™ server. Of course, further users may also be permitted to share and change the first user's calendars. For example, Figure 2 further shows the second desktop computer 110 connected to a local area network (LAN) server 141, which is in turn connected to further desktop computers 115 and 120. The users of the further desktop computers 115 and 120 are able to view and edit the same calendar by means of interconnections with the second desktop computer 110 via the LAN server 141. Yet further users may access the shared calendars by means of the .Mac™ server 131, the RendezVous™ network 135 or yet another server. Indeed, the first laptop 101 may also be connected to the LAN server 141.

[0090] Accordingly, the possible sources of change to a single event in the user's calendar stored on the first laptop 101 include a change from the .Mac™ server 101, a change from the RendezVous™ network 135, a change from another generic server such as LAN server 140, a local change made by the user on the laptop, and a synchronisation operation occurring when a user synchronises the data on his devices, such as another laptop computer (not shown), a PDA (not shown) or a mobile phone (not shown). In addition, an invitation sent by another user may create an event. Clearly, in this situation the use of dedicated calendar server software is undesirable, if not unfeasible.

[0091] Moreover, changes to a calendar may be effected when some or all the relevant users are offline. As the users go online, the change is disseminated through the networks until all users sharing the calendar are correctly updated. Thus, there are many possible sources of change to events in each calendar and these changes can be made asynchronously – that is, changes may be made offline and without knowledge of other changes.

[0092] To achieve this functionality, the present invention considers each calendar as a collection of objects, each object being an event. The event comprises parameter data relating to the event. For example, the parameter data may comprise the start and end times of the event. These times will include the date of the event. The parameter data may also include a description of the event (for example, "dentist"), a status of the event (for example, "very important"), how many attendees there will be at the event and whether the event is to be repeated. In addition, the parameter data may include an indication of whether an alarm is to be sounded and, if so, how long before the event is due to start the alarm should be sounded.

[0093] Unlike the prior art, in which each calendar is stored as a single file, the present invention stores each calendar (or collection of objects) in a folder. When a new event such as a dentist appointment is created, the event is transmitted to

all permitted sharers of the calendar. When the event is transmitted, all parameter data of the event are sent.

[0094] When a user makes a change to the event on his computer, a new version of the event is created and stored. The new version includes all the same categories of parameter data as the original event. However, some of this parameter data will have been changed. For example, Figure 1 shows a dentist appointment between 1:00 PM and 2:00 PM. If the time of the appointment is changed so it is between 12:00 PM and 1:00 PM, the start time and end time parameter data will be changed in the new version but the remaining parameter data will be unchanged. When the user confirms the change, the new version with the new start and end time parameter data will be transmitted to all permitted sharers.

[0095] In the prior art, each previous version of the event is replaced or overwritten in the computer by the new version. This is illustrated in Figure 3A, which shows that an event comprising a meeting at 8:00 AM with Jean Marie was initially created. This was subsequently changed to, and overwritten by, consecutively the same meeting at 12:00 PM; the same meeting at 12:00 PM with an alarm 1 minute before; and the same meeting at 12:00 PM with an alarm 10 minutes before. This last change forms the event and is sent in a format appropriate to the communication protocol required for communication between the computers used to share the calendar.

[0096] However, as shown by Figure 3B, in the present invention all versions of the event are retained by each computer on which the calendar is shared. Thus, the computer uses all versions of the event to establish how the event should be treated. To allow the computers used to share the calendar to distinguish between different versions of an event, the computer on which the change is made further provides the new version with additional data that relates to the change of the event. This additional data may be termed metadata and preferably includes the time at which the change was made or the new version created. The metadata

preferably also includes an identification of the previous version of the event from which the current version was created, an identification of the user who made the change, a description of the change and a user comment relating to the change.

[0097] Effectively then, an event comprises a set of versions. Each version comprises at least some metadata that will be useful to the computers used to share the calendar, in particular to deduce how to treat the different versions of the event and how to display the event. However, there is no requirement for the versions to be ordered. Thus, the communication protocol used to share the calendar between computers can be metadata agnostic. This has the significant advantage that the calendar data need not be stored in any specific format and no specific calendar server software need be provided. Instead, any server software that is capable of storing different versions of the same object can be used to communicate data between calendar-sharing computers. Alternatively, computers are able to share calendar data peer-to-peer, without the intervention of a server at all.

[0098] Figure 4 shows how an event 200 can be conceived as a sack of unordered versions 210. Specifically, Va, Vb, Vc, Vd and Ve are simply different versions of event V and are stored entirely unordered in the communication protocol. In addition, each version will include additional metadata. For example, assume the metadata is a list of the previous versions already present in the event 200 when a new version is created. In addition, suppose that the notation Va (Vc, Vd) signifies that when Va was created, Vc and Vd were already present in the event 200. With the following information:

Vb (Ve, Va, Vc, Vd)

Ve (Va, Vc, Vd)

Va (Vc, Vd)

Vc (Vd)

Vd ()

the computer is able to establish an ordered list of versions, as follows:

$Vd \rightarrow Vc \rightarrow Va \rightarrow Ve \rightarrow Vb$

with Vb being the most recent state of the event. In particular, the latest version Vb can be displayed as the current state of the event to a user. As noted above, the communication protocol for sharing information between calendar-sharing computers pays no attention to the metadata contained in the different versions. This information is used effectively only by the calendar-sharing software provided on each calendar-sharing computer. From the point of view of the sharing protocol, a version is simply a version that is associated with a unique identifier. In the example above, the unique identifiers are a, b, c, d and e. When a new version is received by a calendar-sharing computer, it is simply added to the relevant event, with exactly the same status as the other existing versions for this event. So far as the communication protocol is concerned, the task concerning the new version has been completed. However, the calendar-sharing software provided on the receiving computer must use the metadata associated with the new version to establish how to treat it in relation to the other versions.

[0099] When an event needs to be synchronised between different calendar-sharing computers, or between a calendar-sharing computer and a server, the two devices compare what versions they hold for each event, as shown in Figure 5A. They then transmit to one another the versions that they do not have in common, as shown in Figure 5B, so that both devices have all the same versions, as shown in Figure 5C. Where the device is a calendar-sharing computer, its calendar-sharing software will use the metadata to establish how to treat the event. Thus, where two calendar-sharing computers store the same versions of an event, they

will treat that event the same. Where one of the devices is a server, it will simply act as a passive repository of data and store all versions without regard to the metadata.

[00100] In the foregoing example, the metadata signifies what versions of an event already existed when a new version was created. However, where one or more of the versions were created offline, this may not be sufficient data for a calendar-sharing computer to create an ordered list of versions. To overcome this problem, the metadata may instead or additionally include a time stamp indicating when the event was changed and the version created.

[00101] Figure 6 shows another example of a network in which the present invention may be used. The network shown in Figure 6 is similar to the network shown in Figure 2 and like components have like reference numbers. However, the network shown in Figure 6 does not include the third sub-network 140. Hence, LAN server 141 and additional desktop computers 115 and 120 are omitted. Again, a first user uses the laptop 106, which communicates with the second user's desktop 105 by means of the .Mac™ server 131 and with the third user's desktop 110 by means of the RendezVous™ network 135.

[00102] In step 1 of Figure 7, the first user creates the first version V_0 of an event V. When the first user commits to creating that event, the created (or first) version V_0 is transmitted to .Mac™ server 131 and, when the second user goes online, from .Mac™ server 131 to the second user's desktop 105. At the same time, when the third user's desktop 110 and the first user's laptop 101 are networked, version V_0 is received by the third user's laptop 110. Accordingly, all three computers hold version V_0. This is shown in step 2 of Figure 7. All users then go offline.

[00103] In step 3, the second user changes the event and creates version V_1. Since none of the users is online, only the second user's desktop 105 holds the new version V_1. When the second user goes online, version V_1 is uploaded to the .Mac™ server 131, but will not be sent to the first user's laptop 101 until he

goes online by connecting to the .Mac™ server 131. When he does so, the new version is sent to laptop 101, as shown at step 4. However, at this stage there is no RendezVous™ network connection between laptop 101 and desktop 110 and desktop 110 therefore still does not hold version V_1.

[00104] In step 5, while still offline, the third user changes the original event independently of the change already made by the second user, thereby creating version V_1'. When the third user goes online by forming a wireless network connection between desktop 110 and laptop 101, the two computers compare the versions of the event that they hold respectively. This results in the new version V_1' being sent to the laptop 101 and the previous version V_1 being sent to the desktop 110. Similarly, when the laptop 101 connects to the .Mac™ server 131 they compare the versions of the event that they hold and the new version V_1' is sent to the .Mac™ server 131. Finally, when the second user goes online the desktop 105 receives the new version V_1' from the .Mac™ server 131. Thus, as shown in step 6 all computers eventually hold all versions of the event, although they are received in a different order.

[00105] As illustrated in step 7, the calendar-sharing software on each computer uses the metadata for each version to determine how to deal with the different versions of the event. In this example, the metadata of each version includes a time stamp of when the version was created. Since it was created before version V_1', the software on the third user's desktop 110 uses this data to "insert" version V_1 between versions V_0 and V_1' and determine how and what data is displayed for the event accordingly.

[00106] Another example is shown in Figure 8. In step 1, the first user again creates the first version W_0 of an event W. When the first user commits to creating that event, the created (or first) version W_0 is transmitted to .Mac™ server 131 and, when the second user goes online, from .Mac™ server 131 to the second user's desktop 105. At the same time, when the third user's desktop 110 and the first user's laptop 101 are networked, version W_0 is received by the third

user's desktop 110. Accordingly, all three computers hold version W_0. This is shown in step 2 of Figure 8. All users then go offline.

[00107] In step 3, the second user changes the event and creates version W_1. Since none of the users is online, only the second user's desktop 105 holds the new version W_1. When the second user goes online, version W_1 is uploaded to the .Mac™ server 131, but will not be sent to the first user's laptop 101 until he goes online by connecting to the .Mac™ server 131.

[00108] Before he does so, the third user changes the original version W_0 of the event independently of the change already made by the second user, thereby creating W_1' (step 4). At this stage, the desktops 105, 110 are unaware of the new version the other has created and the first user's laptop 101 is unaware of either of the new versions.

[00109] When the third user's desktop 110 forms a wireless network connection with laptop 101, the two computers compare the respective versions of the event that they hold. This results in the new version W_1' being sent to the laptop 101, as shown at step 5. However, at this stage there is no network connection between laptop 101 and desktop 105 and desktop 105 therefore still does not hold version W_1'.

[00110] In step 6, the laptop 101 connects to the .Mac™ server 131 and they compare the versions of the event that they hold. Thus, version W_1' is sent to the .Mac™ server 131. Finally, when the second user goes online the desktop 105 receives the new version W_1' from the .Mac™ server 131. In addition, the .Mac™ server has received the previous version W_1 from the desktop 105 and sends it to the laptop 101. Thus, as shown in step 6, all computers eventually hold all versions of the event.

[00111] As illustrated in step 7, the calendar-sharing software on each computer uses the metadata for each version to determine how to deal with the different versions of the event. In this example, the metadata of each version again includes a time stamp of when the version was created. Since it was

created before version W_1', the software on the third user's desktop 110 uses this data to "insert" version W_1 between versions W_0 and W_1' and determine how and what data is displayed for the event accordingly. Similarly, the software on the second user's desktop 110 uses this data to "tack on" version W_1' after versions W_0 and W_1, so that the same data is displayed in the same way for the event. Consequently, irrespective of the order in which the computers go online and communicate with one another, the software on each computer effects the display of the same data for each event to each user.

[00112] The metadata can be useful not only for presenting changes to users in the correct order. In a preferred embodiment of the present invention, the metadata also includes an indication of the user who made the change, the time at which the change was made, a description of the change and an optional comment added by the user who made the change. This metadata is then displayed to each user in the form of a history of the event.

[00113] Figure 9 shows an enlarged view of the inspector window 30, which allows the parameters of a particular event in a calendar to be viewed, and the history window 40 of a UI 10 equivalent to that shown in Figure 1. The inspector window 30 shows the most up-to-date parameter data for the event – that is, the parameter data of the most recently created version of the event. In particular, the inspector window 30 shows that an event entitled "Meeting with JMH" is scheduled to start at 2:00 PM on February 26, 2004 and to end at 3:00 PM the same day. The event is included in John's calendar, there are no attendees, there is no status, the event is not repeated and no alarms are set.

[00114] In contrast, the history window 40 shows the metadata for all versions of the event. In this example, the calendar belongs to John (Smith) and a meeting for between 1:30 PM and 3:00 PM was created as an event by his assistant Clara at 3:23PM. The history window in Figure 9 is the window shown to a user logged on to the computer as Clara. Accordingly, the history window shows that the event was created with the comment that "Jean-Marie wants to talk

about iCal Sharing with you". Since Clara created the event, the history window does not show her, as the user, the time or creator of the event. However, other embodiments of the present invention may do so.

[00115] The history window in Figure 9 also shows that John Smith changed the event at 3:37 PM. In particular, he changed the start time of the event to 2:00 PM and added the comment "I'm not available before 2:00 PM. Check with JM if it's OK, please. Thanks." Viewing the full screen shot of the UI 10 in Figure 1, it is clear that the time of the JMH meeting was changed due to a dentist appointment between 1:00 PM and 2:00 PM. The dentist event could have been added after the JMH meeting event was created; Clara might not have had sharing access to the personal calendar in which the dentist appointment event is held; or Clara might have omitted to check the private calendar before creating the JMH meeting event.

[00116] Finally, the history window shows that after checking that the change was acceptable to Jean-Marie, Clara has simply added a message to the event that "Jean-Marie is OK. No problem." Although no change to the parameter data was made, a new version of the event was created with the metadata that the new version was created by Clara, the time of creation and the additional comment. This new version was transmitted to all computers that share John Smith's calendar.

[00117] Of course, the calendar main view 20 in the UI 10 will also show the event, but in the context of a day/date array. In particular, the main view 20 in Figure 1 shows the meeting with JMH on Thu, Feb 26 at 2:00 PM after the dentist appointment. Although, the calendar main view shown in Figure 1 shows a five-day view in day-based linear layout, the number of days in the view 20 can be changed. Alternatively, a grid layout may be used. In this example, all the events in the main view are shown as opaque, with the dentist appointment in the private calendar being shown in a darker colour than the appointments in the work calendar.

[00118] Further examples of history windows are shown in Figures 10A and B. In particular, Figures 10A and B show how the history window 40 viewed by John Smith changes in the above example. Specifically, in the window shown in Figure 10A, John Smith is able to see that Clara has booked the appointment. He then changes the appointment to start at 2:00 PM and adds the comment "I'm not available before 2:00 PM. Check with JM if it's OK, please. Thanks." in the comment box at the bottom. At this stage, the change is local and is not sent to other calendar-sharing computers. As soon as John Smith confirms the modification to the appointment, the metadata of the new event is added in the history box 42 in the history window 40. Thus, in Figure 10B as the registered user John Smith is able to see in the history box 42 that "Me" (that is, he himself) changed the event at 3.37 PM, how he changed it and the comment that he appended. The comment box 41 is now clear.

[00119] A further advantage of the present invention is that the data structure allows a simple notification of users when an event has been changed by another user. In particular, as soon as a user's computer receives a new version of an event (or the first version of an event), it displays an indication of this in the notification window 50 shown in Figure 1. An enlarged view of this window is shown in Figure 11.

[00120] More specifically, as soon as the computer receives a new version of an event it can use the metadata to provide the user with a notification of the change. In the example shown in Figure 11, the calendar-sharing program uses the metadata and the parameter data to display in the notification window 50 the title of the event and either the change that was made to the event or the comment appended to the changed event. Thus, the user is able to see that the events "Pro Apps Review" and "Meeting with Sina" have recently been created and the event "Meeting with JMH" has recently been modified. The bullet point to the left of each title indicates whether the user has already taken into account the

modification. At the user's choice, audio, alternative visual and other alarms may be used to notify the user.

[00121] Of course, the notification window 50 may use a combination of any of the parameter data and metadata in the display.

[00122] In a further embodiment of the present invention, a notifications filter may be provided. This allows the user to select the events or calendars for which he requires notifications of a change (or for which he does not require notifications of a change). For example, imagine that an office administrator holds a separate calendar for each of ten meeting rooms available in an office. He shares those calendars with all members of staff, who are able to book time slots by creating an event in the appropriate calendar at the appropriate time. However, most members of staff will not require to be notified each time one employee books a time-slot in a meeting room and thereby creates a change in the calendar. The notification filter can be provided to filter out all notifications to a user of changes to the meeting room calendars unless he requests them. However, if a user has booked a time-slot on a meeting room calendar (and therefore created an event), he can set the notification filter to notify him only of modifications to that event – that is, only when a new version of that time-slot is received by the user's computer.

[00123] It will be clear from the foregoing description that the calendar-sharing data protocol of the present invention is essentially an "add only" protocol. In other words, all versions of an event are retained and none are deleted or destroyed during normal operation. This has the significant benefit that no dedicated calendar server software is required. Instead, all the "intelligence" is held by the computer programmed with the calendar-sharing software and the server, if used, merely acts as an "unintelligent" repository of the calendar data. More specifically, the computer programmed with the calendar-sharing software provides the intelligence to add metadata to each new version of an event and the intelligence to interpret a plurality of different versions of the

same event and to display them meaningfully to the user. When a new version arrives in the computer programmed with the calendar-sharing software, it is simply added to the event, with exactly the same status as the other versions of the event. From the point of view of the sharing protocol, the task concerning the new version is completed. Of course the calendar-sharing software must then interpret the metadata to determine how the event should be displayed.

[00124] Further benefits of the protocol provided by the present include allowing a history of an event to be easily maintained and changes in an event to be notified to users. Moreover, the protocol is very versatile and highly adaptable. In particular, the protocol allows changes to an event in a calendar to be made using the local device on which the calendar is held, a .Mac™ server, a RendezVous™ network and any other suitable versioning server (such as a DeltaV server acting for a LAN). By a versioning server is meant any server that is capable of storing several versions of the same object. Generally, versioning servers store the different versions of an object in a particular order. However, in the present invention the different versions can be stored in any order, like a sack of potatoes. Direct data transfer between computers and other devices, for example by physical wiring, telephone communications, Bluetooth and other IR communication, are also possible.

[00125] As noted above, the sharing protocol of the present invention involves an "add only" rule. Accordingly, events are not deleted in the present invention. Rather, when a user wishes to delete an event, a new version of that event is created, the new version being marked "deleted". One example of how a deletion may be handled by the present invention is shown in Figure 12.

[00126] Similar to Figures 7 and 8, user 1 creates an event X_{00} (step 1) and sends it to users 2 and 3 (step 2). For the purposes of illustration, each user displays the event as rectangle in Figure 12. In step 3, user 2 decides to delete the event. Accordingly, a new version of the event X_x is created and user 2 no longer displays a rectangle.

[00127] In step 4, users 1 and 3 receive the new version X_x. Since they are not the creators of new version X_x, they display the event in a format indicating that it has been deleted. In Figure 12, users 1 and 3 display the event as a rectangle with a cross through it. However, in the calendar main window 20, the "deleted" event could be shown in a number of ways. For example, it could be shown in red or another colour, or as translucent, or as a combination. Other suitable forms of display will be apparent to those skilled in the art.

[00128] Users 1 and 3 then have three options to decide how to treat the deletion by user 2. First, they can simply ignore the deletion and no further steps are taken. Second, they can choose to accept the deletion. Thus, step 5 shows that user 3 has accepted the deletion and no longer displays the rectangle. Effectively, the computer has been instructed to treat version X_X by no longer displaying the event. Analogously, user 3's calendar main window 20 may no longer show the event. Note that users 2 and 3 still retain versions X_{00} and X_X but simply do not display them.

[00129] The third option is to reinstate the event. This is shown in step 6, in which user 1 has copied all the parameter data of version X_{00} to create version X_{01}. However, version X_{01} will have its own metadata, indicating who made the change, when and how. In step 7, the most recent version, X_{01}, is transmitted to users 2 and 3, which subsequently display the event.

[00130] In one embodiment, to save space or permanently prevent an event from being viewed, only the owner of the calendar may cause the event to be physically deleted. This option cannot be undone and involves removing all versions of the event and replacing the event everywhere with a token indicating that the event is now deleted and read-only. This is particularly important where a server is used to share the calendar data.

[00131] In addition, in a preferred embodiment users are able to move events from one calendar to another. Preferably, this entails copying an event in a first calendar, marking the event deleted in the first calendar and pasting the

copied event to create a new event in a second calendar. The step of copying may involve copying all versions of the event, including the metadata for each version. In that case, the history of the event will be recoverable in the new calendar. Alternatively, only the most recently created version and its metadata need be copied. A further alternative is to copy only the parameter data but not the metadata of the most recently created version. In that case, no history will be retained for the event in the second calendar. Of course, a new history will be created as the event is changed in the second calendar. It will be apparent to persons skilled in the art that other options are possible when moving events between calendars.

[00132] In a preferred embodiment, duplication, cut, copy and paste operations within a calendar may also be performed. These steps are similar to the move operation described above. In particular, a cut and paste operation will involve copying the event (and at the user's option all versions of the event so the history is retained), marking the original event as deleted so it is no longer displayed, and pasting the event at a different location.

[00133] Since the sharing protocol of the present invention involves an "add only" rule, there is no way to "undo" a change in the traditional sense. To undo a change in the present invention, a user may select one of the previous versions of the event. The parameter data of that version are then copied to form a new version with new metadata. Of course, the original version and all intervening versions are maintained. Thus, the metadata may cause the history box 42 to display the creation of the new event as an undo.

[00134] Imagine for example that an event Y is created (version 0) with the parameter data (a) and this is changed to version 1 with parameter data (b). A series of changes could be represented as:

$$Y_0(a) \rightarrow Y_1(b) \rightarrow Y_2(c) \rightarrow Y_3(d) \rightarrow Y_4(b).$$

[00135] In this case, after version Y_3 the user wished to undo the last two changes and revert to the event parameters of version 1. Accordingly, the

parameter data (b) of version 1 was copied to form version 4. However, version 4 will have its own metadata and versions 1, 2 and 3 are maintained. Thus, the history box can show all the changes.

[00136] The present invention also allows users to schedule recurring events, such as a weekly meeting. However, as noted above, when a server is used to share calendars in the present invention, the server effectively acts as a passive repository of data. The present invention therefore treats recurring events as single objects involving a recurrence rule. The recurrence rule might involve, for example, setting the start and end dates of a recurring event and the frequency of the recurrence (daily, weekly, monthly, yearly etc).

[00137] Thus, when a recurrence rule changes, the individual events or occurrences are not modified separately. Rather, the whole set of recurring events is changed. However, a change to a recurrence rule may have the effect of modifying a single, specific occurrence in the set of recurrences. When a single, specific occurrence in the set of recurrences is changed, that occurrence may be considered as a detached occurrence.

[00138] Accordingly, in a preferred embodiment of the present invention the history of individual occurrences resulting from a recurrence rule comprises two parts. The first part displays the changes made to the specific selected occurrence or to all occurrences that do not involve a change to the recurrence rule itself. The second part displays the changes to the recurrence rule itself, for example any changes to the start date or end date of the recurrence rule.

[00139] The present invention has been described with particular reference to sharing a calendar between users on different computers. However, the present invention is not limited to the application of calendar sharing. Rather, it can be applied to any data set that can be considered to comprise a set of objects and which can be changed by any of a plurality of users. For example, the present invention could be used to share and edit collections of photographs or other images by user groups. In that case, each image would be considered as a

separate object, the parameters of the object being the image data. Each time an image is edited, a new version would be created with new metadata relating to the change and the history and notification boxes would be updated appropriately. As with the calendar-sharing application discussed above, the history box could show the time the change was made; an indication of the previous version that was changed to create the new version; an indication of what change was made; and an indication of who made the change.

[00140] In another embodiment, the present invention allows a collaborative document to be authored by several people. In that case, the document can be considered as a collection of paragraphs (or chapters and/or other sections such as images, as determined by the users), the parameter data relating to the text and formatting of each paragraph or other section and the metadata relating to the time the change was made; an indication of the previous version that was changed to create the new version; an indication of what change was made; and an indication of who made the change.

[00141] Referring to Figure 13, the present invention differs from the prior art by separating the conduits 2008 and 2010 from the synchronisation engine 2006. Moreover, synchronisation software is provided for each device conduit rather than having a single synchronisation software for operating the synchronisation engine 2006. In this manner, the present invention overcomes the fifth problem discussed above, namely the synchronisation engine having synchronisation software which is applicable to all devices. Accordingly, each conduit is able to function independently of any other conduit. Some devices may not be able to support separate synchronisation software and have it's own conduit such as that shown as device 2002. None the less, the synchronisation method accordingly to the present invention enables such devices to be accommodated.

[00142] The present invention also differs from the prior art in that the synchronisation engine 2006 now includes a truth table 2020. The truth table is

an amalgamated copy of the records from all of the devices involved in the synchronisation system. Thus, during synchronisation, each device is synchronised serially one at the time with the truth table and each record of the device is synchronised with each record in the truth table. Having obtained an amalgamation of all of the updated records from all of the devices, only then are the devices synchronised with the truth table. The devices, according to the synchronisation system of the present invention, are never directly synchronised with each other but only with the truth table.

[00143] Since the devices are each synchronised with the truth table, this simplifies the first problem enumerated above, when devices are absent. Moreover, the present invention also obviates the second problem by avoiding redundant synchronisation steps since the same change being submitted by two devices is only applied once to update the truth table. The system in the present invention also obviates the fifth problem, in that since the devices are each updated in turn, the truth table is not accessed simultaneously and so conflicts are avoided by the devices being synchronised simultaneously.

[00144] The truth table is defined not by the number of changes but rather by the number of records held on all the devices. Accordingly, the truth table is defined by the total number of records. This is in contrast to the prior art which effects synchronisation by storing the total number of changes. The truth table can provide data for updating devices depending upon the requirements of the devices. In some cases the devices merely want the deltas whereas some devices require the whole record.

[00145] As in the prior art, the present invention enables conflicts to be resolved. In the example given in Figure 16 (which omits features not of relevance for this part of the description), device 1 submits a deletion of record number 7, an add of record number 2 and a modify of record number 1. This is applied to the truth table 2020 by a mingler 2016 in the synchronisation engine 2006. Device number 2 submits a modified to record number 9, a modified to

record number 3 and an add to record number 2. This is also applied to truth table 2020. Device number 3 then submits changes which involve delete of record number 9, a different modify to record number 3 and a modify of record number 5. The mingler 2016 calls for conflict resolution, those conflicts involving record number 9 and record number 3, and the result of that conflict resolution is stored in truth table 2020. The truth table then contains an aggregate of all of the changes involved on the three devices.

[00146] Each of the devices are then in turn updated so as to be synchronised with the truth tabled 2020 but omitting the changes which are submitted by that device. Hence, device number 1 does not require the data involving the deletion of record number 7, the addition of record number 2 and modification of record number 1. Instead, the conduit for device number 1 extracts from the truth table 2020 the changes to be updated, namely the deletion of record number 9, the modification of record number 5, the alternative modification of record number 3 and the addition of record number 2. Similar updates are also obtained and effected by the respective conduits for devices 2 and 3.

[00147] As discussed previously, some data to be synchronised involves not only attribute data but also relationship data. It is known to model data using a form of the entity relationship model (ERM). This enables the data to be categorised into records and relationships between the records. The data is categorised by a schema 2022. The present invention includes a schema 2022 in the synchronisation engine 2006. Since the schema categorises the data into records and relationships, it is able to define and vary the definitions of the data categorisation. Hence, the schema together with the details of the device capabilities provided by the device specific areas 2008a and 2010a, the synchronisation engine can accommodate for truncation or translation of the data by any one of the devices. For example, consider the following data held by two devices:

Field Name	Device 1	Device 2
First	Steve	Steve
Middle	G	
Last	Smith	Smith

[00148] In this example, Device 2 does not retain the middle field. Hence, the schema identifies certain fields as an identity key. If, the schema identifies the first and last name as identity keys, then the records held by device number 1 and device 2 will be considered to be same. The use of a schema in the synchronisation engine is particularly useful in overcoming the third problem enumerated above.

[00149] Another such example is when a contact list includes all details of a particular person, as discussed above. However, on a device such as a mobile phone, only the home, work and mobile telephone numbers are required and not any of the addresses. Thus, the schema would define the data from such a mobile telephone as only comprising those telephone number fields.

[00150] It is to be noted that, in contrast to the prior art, although the synchronisation method may be effected through the user interface, the present invention more preferably is initiated by the device or application itself depending upon the criteria set for that device or application.

[00151] The synchronisation method according to the present invention involves four phases. These phases include negotiation, application, mingling and updating, and these are discussed in more as follows:

[00152] Negotiation

[00153] In the first instance, each conduit must negotiate the synchronisation mode. As noted above, there are two types of synchronisation. Normally, the synchronisation mode selected is that of fast synchronisation. However, some devices may not be able to support a fast synchronisation, or indeed the conduit may not be able to select the relevant records for fast

synchronisation and so elect to proceed with slow synchronisation. The synchronisation engine then confirms which synchronisation mode is to be effected and, accordingly, the conduit interrogates the device according to the appropriate mode of synchronisation.

[00154] Application

[00155] Once the synchronisation mode has been negotiated, the conduit extracts the changes from the device when undergoing fast synchronisation. When the synchronisation mode involves slow synchronisation, all data to be synchronised is extracted by the conduit and passed to the synchronisation engine 2006.

[00156] Mingling

[00157] The mingler receives the changes from all of the conduits and applies those changes first in turn from each device and then through each record. Any conflicts between changes are identified. The changes are then applied to the truth table.

[00158] If there is a conflict with any record, the synchronisation engine first tries to resolve the conflict using a set of rules specific to the record in question. If a conduit has added customised field to a record type, then the conduit specific to that device may attempt to resolve the conflict. Only if the conflict cannot be resolved using such rules, will the synchronisation engine then request resolution from the user.

[00159] The step of mingling also involves optimising a set of consecutive changes to a record by discarding all but the final change. For example, if one device changes a field in a record from value A to B and then on a subsequent synchronisation from value B to C, then the mingler optimises the changes by applying only the change from A to C. This change from A to C is then applied to update any devices required.

[00160] Updating

[00161] The final step in the synchronisation process is for each conduit to receive from the synchroniser the changes stored in the truth table and prepare those for updating the respective device. If the device requires any truncation or translation of the data, then the conduit stores that truncation rule in the store. Having effected the updating of the device with all of the changes, then the conduit confirms that the updating has been completed to the synchronisation engine.

[00162] As noted above, it can be a problem when devices truncate or translate data stored on that device. In addition to providing full flexibility for the schema in the synchronisation engine, the synchronisation method also differs from the prior art by providing a more efficient solution to this problem of truncation or translation of data. The synchronisation method thus enables the conduit to compare fields between the device and the conduit store to identify whether there are any changes. If the two fields match, then no change has been effected and the conduit need not advise the synchronisation engine in relation to that field. However, the fields may differ between that stored in the store and that stored in the device. As in the prior art, the conduit through the device specific areas 2008a and 2010a seek to extract the full record from the device and the store together with any truncation or translation rules which may be applied. The conduit then compares the two full records taking into account any truncation or translation rules. The present invention differs from the prior art in that the conduit also considers what each record or field might look like:

a) from the device;

b) to the device; and

c) and when actually compared with each other

this is known as the triple comparison test and enables fields or records to be compared to the device, from the device and the actual field or record. This significantly reduces the number of conflicts that are passed for conflict resolution.

[00163] The method of synchronising according to the present invention also includes a solution to the problem of poor synchronisation of relationships. This is achieved through providing the schema to be able to define more flexibly the data categorisations and in addition whether fields are connected or dependent upon each other and the type of dependency.

[00164] The schema also acknowledges and tries to preserve the order of changes. There are various modes of ordering and these are as follows :

- the fact that there is no order;
- weakly ordered : which implies the orders specified are preferable but not necessary and so if any conflict arises, then the synchronisation should attempt to resolve the conflict without seeking conflict resolution;
- strongly ordered : where the order is considered necessary and so if there is any conflict, then the matter should always be passed for conflict resolution.

[00165] Through the use of acknowledging and preserving the orders, this enables the synchronisation of relationships to be preserved.

[00166] An example of such ordering is as follows:

Truth table		Conduit store		Device
A B C	→	A B C	→	A C
A" B C	←	A B C	←	A" C
A" B C D	→	A" B C D	→	A" C D

[00167] In this example, the truth table contains records or fields A B and C. These are also stored in the conduit store but with B noted as being not supported by it's respective device. Thus the device only stores A and C. A change is made to A by the device and this is compared by the conduit with that stored in the conduit store. The comparison does not involve B since the conduit store confirms that B is not supported by the device. The change is passed into the truth table. In this case, the absence of B in the device is only indicated as weakly ordered and so it is not passed for conflict resolution. Subsequently, the

truth table is updated from another device by the addition of D. This is passed through the conduit to update the store and the device. Again, no conflict is raised due to the weak ordering.

[00168] Thus, when synchronising such a device, the following information would be present in the truth table, conduit store and device.

[00169] As noted above, the present invention is particularly directed to overcoming five problems and these include accommodating for absent devices or applications, avoiding redundant synchronisation steps, accommodating for truncation or translation of data by devices or applications, enabling and preserving synchronisation of relationships and obviating known synchronising methods wherein two devices or applications are accessing the same database thereby causing inherent conflict. The present invention provides the solution for each of those problems as discussed above. In addition, the synchronisation method effects what is known in the art as trickle synchronisation. This enables each device to effect synchronisation frequently. Hence, only a small amount of data at any one time is held in the truth table. This results in faster synchronisation and involves less conflict. Whenever conflicts do arise, the user through the user interface may resolve those conflicts at that time or elect to resolve them at a later date.

[00170] In order to effect trickle synchronisation more efficiently, the user also defines the mode of synchronisation for each of the devices involved in the system. Typically, if the device involves a computer application, then fast synchronisation is elected. If the devices involve connection to a PC through Wire, Fire Wire or Blue Tooth, then it is usual to elect slow synchronisation since almost certainly all data will need to be submitted for such synchronisation. Where the device comprises a server, it involves low latency but high connectivity and therefore it is usual to elect slow synchronisation. Thus, each of the types of devices may elect the type of synchronisation and, moreover, may elect when such synchronisation is effected. For example, if the device is a

mobile phone, whenever the phone is in range, then the synchronisation process may be effected. Alternatively, if the device comprises a computer program for managing calendars, then whenever the program is initiated, then it is usual to instruct synchronisation to be effected first.

[00171] By allowing the synchronisation method of the present invention to effect trickle synchronisation, each of the devices or applications is synchronised optimally. The present invention relies upon the use of a truth table. However, that truth table may be stored on not just one device but on more than one device. Figure 17 illustrates two principal devices, 2024 and 2026, each of which contains a truth table. Each device 2024 and 2026 has a number of satellite devices 2028 with whom synchronisation is effected through a respective conduit and synchronisation engine. When the two devices 2024 and 2026 are connected either directly or through an intermediary device 2030, such as a server, then there may be different truths in the truth tables stored on devices 2024 and 2026. No one truth table has precedent. However, when the two devices 2024 and 2026 are synchronised, then the changes from one are passed to another and visa-versa. If any conflict occurs, then the synchronisation method assumes that the one initiating the synchronisation is assumed to be the master and its changes in any conflict are passed and implemented in the truth table on the other device.

[00172] The present invention relates to a method of synchronising which includes not only storing the actual data which has been changed but also together with logging the synchronisation events and the devices involved in those events. Hence, as shown in Figure 19, data is acquired in an Administration Table indicating which devices are involved in the synchronisation system. In this case, the devices indicated are a phone, palm device, home computer and work computer. The synchronisation events are indicated in the column generation. In this case, the synchronisation system has been initialised such that there have

been no synchronisation events. Accordingly, for each device, the generation indicated is zero.

[00173] Figure 19 also illustrates the Truth Table which provides a format for storing the data being changed under the column Datum. The synchronisation events are stored in one or two columns, Modify Generation and Add Generation. Modify Generation events involve where the field or record has already existed and the change involves a modification of that data. Conversely, Add Generation notes those new fields or records to be added. Finally, the Truth Table also indicates which device or client has provided the change under the column Client. Since Figure 19 refers to the initialised state of the synchronisation system, there is no data in the Truth Table.

[00174] Figure 20 demonstrates the type of data which may be stored after the first synchronisation event. In this case, the phone in the synchronisation system is the only one involved in the synchronisation event. The phone submits two new records or fields X and Y. Hence, in the Administration Table, the device or client phone is indicated as being involved in the first synchronisation event. In the Truth Table the data X and Y is indicated as being added in synchronisation event 1 by the device or client phone.

[00175] In Figure 21, the phone also instigates the second synchronisation event by adding datum Z. Hence, the Administration Table is updated to indicate that the phone is involved in the second synchronisation event, and that the Truth Table has added Datum Z in the second generation. In this case, it is necessary to retain both the previously added datums X and Y from the first synchronisation event since the other devices or clients in the system, the palm device, home computer and work computer, have not been advised of the changes involved in the first synchronisation event.

[00176] In Figure 22, all of the devices or clients are present during the third synchronisation event. Hence, the Administration Table indicates all of the clients as being present in the third synchronisation event. In the third

synchronising event, no data changes have occurred. Hence, no additional datum is indicated in the Truth Table. However, since the palm device, home computer and work computer were not present during the first and second synchronisation events, these devices are updated from the Truth Table with the changes to the X, Y and Z datum. Since the phone submitted the datum changes X, Y and Z during the first and second synchronisation events, there is no need to update the phone with the datum X, Y and Z.

[00177] Since all of the clients were present during the third synchronisation event and that all of the clients were updated with all of the changes indicated in the Truth Table, then it is possible to delete those changes from the Truth Table. This is shown in Figure 23. The Administration Table logs that each of the clients were each last involved in the third synchronisation event but since all of the devices were updated with all of the changes up to the third synchronisation event, those changes may be deleted from the Truth Table and the Truth Table is blank.

[00178] Figure 24 illustrates the fourth synchronisation event, whereby the phone adds a new record A.

[00179] Figure 25 illustrates the palm device in the fifth synchronisation event, adding a new record B. The palm device, since it was not involved in the fourth synchronisation event, is updated with the new datum A which was submitted by the phone in the fourth synchronisation event. However, the palm device is not updated with datum B since the palm device submitted the datum in the fifth synchronisation event. Moreover, the phone is not updated with datum B since it was last present during the fourth synchronisation event and is not present for the fifth synchronisation event. Accordingly, the phone includes the new datum A but not the new datum B. The palm device includes both datum A and B. The home and work computers include neither datum A or B.

[00180] In Figure 26, the phone in the sixth synchronisation event adds the new datum C. Moreover, the phone is updated with the new datum B submitted in the fifth synchronisation event from the palm device.

[00181] In Figure 27, the phone, palm device and home computer are all present during the seventh synchronisation event which adds datum D on the home computer. Since the phone was present during the sixth synchronisation event, the phone only needs to be updated with datum D. The palm device was previously present during the fifth synchronisation event and so will be updated with datum C and D. The home computer, since it was only previously present during the third synchronisation event, is updated with datum A, B and C. However, the home computer is not updated with datum D since the home computer submitted datum D itself. Finally, the work computer is not updated since it was not present during the seventh synchronisation event and was previously only present during the third synchronisation event. Accordingly, any changes since the third synchronisation event must be retained in the Truth Table.

[00182] In Figure 28, all devices in the synchronisation system are present during the eighth synchronisation event. Accordingly, the work computer is updated with datum A, B, C and D. Since no further datum changes were submitted by any of the devices, the Truth Table can then be cleared of such changes. This is shown in Figure 28.

[00183] In Figure 29, during the ninth synchronisation event, the phone submits a modification of datum G'.

[00184] In Figure 30, the work computer submits a further modification to datum G". The datum G is thus modified by two different devices. Accordingly, the synchronisation engine seeks in the first instance to identify whether any conflict exists. Various scenarios may be encountered during such identification:

- The two devices may submit the same change. Thus, if G' equals G", then clearly there is no difference between the datum and hence no conflict exists.

- If the datum comprises a field of a record as opposed to a record *per se*, then G' may relate to one field, whereas G" may relate another field of the same record. Hence, again there would appear to be no conflict.

- If the datum comprises a whole record, then G' will not equal G" so that a conflict will be found to exist. However, the schema may include rules for defining whether such conflicts may be overlooked or whether they should be passed for conflict resolution. Such rules may include whether the datum involves an identity key such that all conflicts submitted to the user for resolution.

[00185] In any case, having identified whether a conflict exists and having resolved that conflict, the Truth Table is then updated with the resolved conflict. Subsequent synchronisation events and updating steps are then effected, as discussed previously.

[00186] As a consequence of storing the actual data changes together with the log of synchronisation events and the devices involved, it is possible to optimally identify only those comparison steps which need to be effected. Moreover, since the base line upon which the synchronisation method relies is the log of synchronisation events and clients involved, any modification of the systems clocks will not affect the optimal efficiency of the synchronisation method.

[00187] The present invention is applicable to any known synchronisation system, device or data. In particular, the synchronisation method according to the present invention is particularly useful to not only attribute data but also relationship data. In the example where the data comprises contact lists, a persons contact details may include home telephone number, work telephone number and mobile telephone number as well as various addresses including e-mail addresses of both work and home, and postal address and work address.

Each of the contact details would be considered a field, whereas all of the contact details for a particular person would be considered the record. Either the field or the record may be considered attribute data. The contact list may also include the relationships between that person and other persons held in the contact list. This could include the fact that the first person is a brother to a second person. A third person's details may also be given together with the relationship that he is a father to both the first and second person. Any types of relationships may be given, not just relative relationships but also relationship information such as girlfriend, boyfriend or partner, work colleague or other contact relationships.

[00188] Figure 31 indicates the type of Truth Table that may be held when synchronising both attribute data and also relationship data. In this case, during synchronisation event 11, the work computer submits modifications to record number 1 in relation to datum fields S" and Q. The home computer during the synchronisation event 12 submits a modification to record number 2, including the datum field T.

[00189] The Relationship Table includes details of the relationship from source record number 1 to target record number 2, to indicate that the relationship is a parent relationship. Conversely, record number 2 is indicated as having a relationship as a child from record number 2 to both record number 1 and record number 3.

[00190] In Figure 31, the Administration Table has not been included to concentrate on the Truth Table. Moreover, the Truth Attribute Table defines the datum as a field level and also includes a record number. This is not necessarily required according to the present invention but does assist in identifying all of the changes to a particular record. In all other respects, the Truth Tables indicated in Figure 31 are merely a modification and extension of the Truth Tables indicated in the previous figures.

[00191] Other implementations of the present invention will be immediately apparent to those skilled in the art.

[00192] The foregoing description has been given by way of example only and it will be appreciated by a person skilled in the art that modifications can be made without departing from the spirit and scope of the present invention.

CLAIMS

We claim:

1. A method of sharing a group of one or more objects between a plurality of users, in which one or more of said plurality of users is able to change parameter data of at least one said object, the method comprising:

storing at least one version of each said object;

when an object is changed, creating a new version of the object, the new version of the object comprising additional data relating to the creation of the new version;

storing the new version of the object together with any version of that object before the change;

providing all versions of the object to each of said plurality of users; and

using the additional data provided for each version of the object to determine how to display the object.

2. A method according to claim 1, wherein the group is a calendar and each object is an event in the calendar.

3. A method according to claim 2, wherein object parameter data comprises at least one of a start time of the event, an end time of the event and a description of the event.

4. A method according to claim 3, wherein the object parameter data further comprises at least one of a status of the event, whether the event is to be repeated and the persons attending the event.

5. A method according to claim 3 or claim 4, wherein the additional data comprises at least one of an identification of the user who made the change, a time at which the change was made, a description of the change, a user comment

relating to the change and an identification of the previous version of the event from which the present version was created.

6. A method according to claim 1, wherein the group is a text document and each object is a section of the text document.

7. A method according to claim 6, wherein object parameter data comprises text data.

8. A method according to claim 7, wherein the additional data comprises at least one of an identification of the user who made the change, the time at which the change was made, a description of the change, a user comment relating to the change and an identification of the previous version of the event from which the present version was created.

9. A method according to claim 1, wherein the additional data comprises the time at which the change was made, the method further comprising displaying the most recently changed version of the object.

10. A method according to claim 1, wherein the additional data comprises an identification of the previous version of the event from which the present version was created, the method further comprising displaying the object in accordance with a sequence of changes established according to the relationships between different versions of events.

11. A method according to claim 1, the method further comprising displaying one version of the object together with additional data of another version of the object.

12. A method according to claim 11, wherein the additional data displayed includes at least one of an identification of the user who made the change, the time at which the change was made, a description of the change, a user comment relating to the change and an identification of the previous version of the object from which the present version was created.

13. A method according to claim 1, the method further comprising sharing the group of objects between the plurality of users by means of user devices in a network.

14. A method according to claim 13, the method further comprising storing all versions of objects in user devices accessed by users.

15. A method according to claim 14, the method further comprising storing all versions of objects in a network server.

16. A method according to any one of claims 13 to 15, wherein a first user device connected in the network compares all versions of the objects it stores in a group with all versions of the objects a second device connected in the network stores in the same group.

17. A method according to claim 16, wherein the first user device receives from the second device versions of objects not stored by the first user device.

18. A method according to claim 16, wherein the first user device transmits to the second device versions of objects not stored by the second device.

19. A method according to claim 1, wherein when a user deletes an object from the group the method comprises:

creating a new version of the object with additional data indicating that the object is deleted,

storing the new version of the object together with any version of the object before the deletion, and

displaying the object with an indication that the object has been deleted.

20. A method according to claim 1 or claim 19, wherein when a predetermined one of said plurality of users deletes an object from the group the method comprises:

deleting all versions of the object and replacing them with a token indicating that the object has been deleted and can no longer be changed.

21. A method according to claim 1, the method further comprising sharing a plurality of groups of objects.

22. A method according to claim 21, the method further comprising moving an object from a first group to a second group.

23. A method according to claim 22, the step of moving an object from a first group to a second group comprising copying all versions of the object in the first group and storing them in the second group.

24. A method according to claim 1, the method further comprising selecting an existing version of an object, copying object parameter data of the selected version and storing the copied object parameter data with new additional data as a new version of the object.

25. A method according to claim 1, further comprising notifying one or more of said users when an object is changed.

26. A method according to claim 1, wherein the group is a collection of images and each object is an image.

27. A computer program for causing a computer to perform the method of claim 1.

28. A recording medium having recorded thereon a computer program for causing a computer to carry out the method of claim 1.

29. A network comprising at least a first user device and a second user device, wherein said user devices are arranged to carry out a method according to claim 1.

30. A network comprising at least a first user device and a second user device, wherein said user devices are programmed with a computer program according to claim 28.

31. A network according to claim 29 or claim 30, further comprising a server.

32. A user device arranged for use in a network according to claim 29 or claim 30, or to carry out the method of claim 1, or programmed with a computer program according to claim 27.

33. A method of synchronising data between a primary device and one or more subsidiary devices, said method comprising:

storing a primary set of data on said primary device;

comparing data on each subsidiary device with said primary set of data;

updating said primary set of data; and

updating data on each of said subsidiary devices using said updated primary set of data.

34. A method of synchronising as claimed in claim 33, further comprising:

resolving any conflicts before updating the primary set of data.

35. A method of synchronising as claimed in claim 34, further comprising:

optimising data changes after resolving any conflicts and before updating the primary set of data.

36. A method of synchronising as claimed in any one of claims 33 to 35, further comprising:

categorising data before storing said primary set of data.

37. A method of synchronising as claimed in claim 36, in which said categorising includes categorising the order between any data.

38. A method of synchronising as claimed in claim 33, further comprising:

translating data from a device format to a primary format before storing said primary set of data and before comparing data from said subsidiary device.

39. A method as claimed in claim 38, further comprising:

translating data from the primary format to the device format before updating said subsidiary devices.

40. A method of synchronising as claimed in claim 38 or 39, further comprising:

storing details of each subsidiary device.

41. A method of synchronising as claimed in claim 40 when dependent upon claim 38, in which said storing details includes storing truncation details and further comprising

reverse truncating said data from said subsidiary device before translating data from the device format to the primary format.

42. A method of synchronising as claimed in claim 40 when dependent upon claim 39, in which said storing details includes storing truncation details and further comprising

truncating said data from said primary device before translating data from the primary format to the device format.

43. A method of synchronising as claimed in claim 40, in which said storing details includes method initiation details and further comprising:

initiating said method of synchronising for said device in accordance with said method initiation details.

44. A method of synchronising as claimed in claim 40, in which said storing details includes synchronisation mode details and further comprising:

effecting said method of synchronising for said device in accordance with said synchronisation mode details.

45. A method of synchronising as claimed in claim 33, in which said comparing data includes three comparisons, comparing data which might be received from the device with the primary set of data, comparing the data which might be sent to the device with the primary set of data and comparing the data actually received from the device with the primary set of data.

46. A method of synchronisation as claimed in claim 33, in which said data comprises attribute and/or relationship data.

47. A method of synchronisation as claimed in claim 33, in which said primary and subsidiary devices includes applications stored on or run by electronic devices.

48. A method of synchronisation as claimed in claim 47, in which said devices comprise applications stored on or run by the same electronic device.

49. A method of synchronisation as claimed in claim 47 or 48, in which said electronic devices include computers, palm devices, personal digital assistants, music devices and mobile telephones.

50. A method of synchronising between three or more devices, said method comprising:

storing an indication of the device or devices involved in each synchronisation event;

storing data changes received during a current synchronisation event together with the device submitting those changes; and

applying the data changes subsequent to the stored synchronisation event for the or each device.

51. A method of synchronising as claimed in claim 50, further comprising identifying data changes prior to storing data changes.

52. A method of synchronising as claimed in claim 50 or 51, in which said applying the data changes does not include the data changes submitted by the device during a current synchronisation event.

53. A method of synchronising as claimed in claim 50, in which said data changes include adding new data, deleting data or adding data.

54. A method of synchronising as claimed in claim 50, in which said data comprises attribute and/or relationship data.

55. A method of synchronising as claimed in claim 50, further comprising: identifying any stored data changes which have been applied to all devices; and deleting said data changes from said store of stored data changes.

56. A method of synchronisation as claimed in claim 50, in which said data comprises attribute and/or relationship data.

57. A method of synchronisation as claimed in claim 50, in which said devices includes applications stored on or run by electronic devices.

58. A method of synchronisation as claimed in claim 57, in which said devices comprise applications stored on or run by the same electronic device.

59. A method of synchronisation as claimed in claim 57 or 58, in which said electronic devices include computers, palm devices, personal digital assistants, music devices and mobile telephones.

Fig.1.

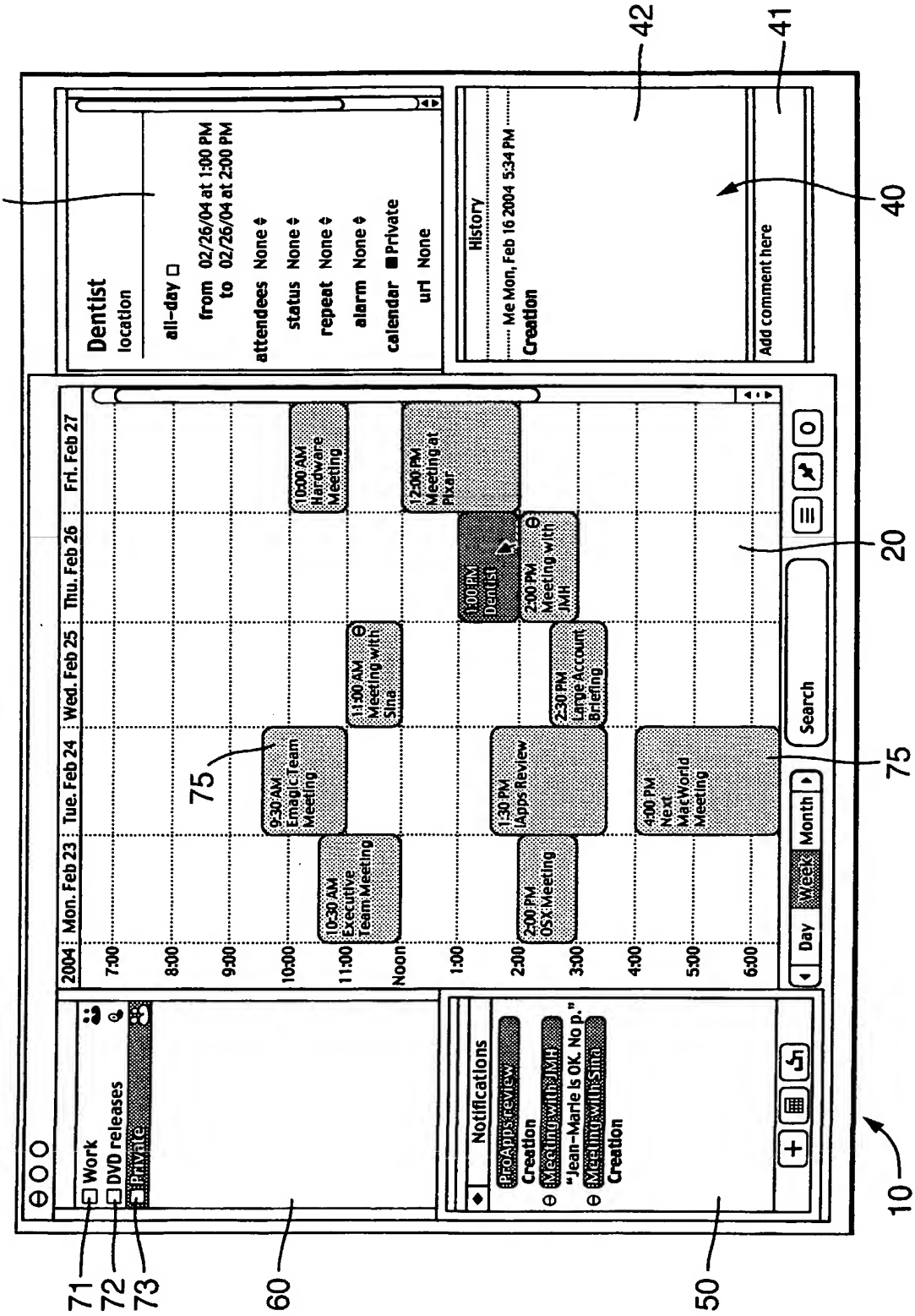

30
42
41
40
Dentist
location
all-day
from 02/26/04 at 1:00 PM
to 02/26/04 at 2:00 PM
attendees None
status None
repeat None
alarm None
calendar Private
url None
History
Me Mon, Feb 16 2004 5:34 PM
Creation
Add comment here
20
75
Search
Day
Week
Month
2004
Mon. Feb 23
Tue. Feb 24
Wed. Feb 25
Thu. Feb 26
Fri. Feb 27
10:30 AM Executive Team Meeting
9:30 AM Emagic Team Meeting
11:00 AM Meeting with Sina
10:00 AM Hardware Meeting
2:00 PM OSX Meeting
1:30 PM iApps Review
2:30 PM Large Account Briefing
12:00 PM Meeting at Pixar
1:00 PM Dentist
2:00 PM Meeting with JMH
4:00 PM Next MacWorld Meeting
Work
DVD releases
Private
Notifications
ProApps review
Creation
Meeting with JMH
"Jean-Marie is OK. No p."
Meeting with Sina
Creation
71
72
73
60
50
10

Fig.2.

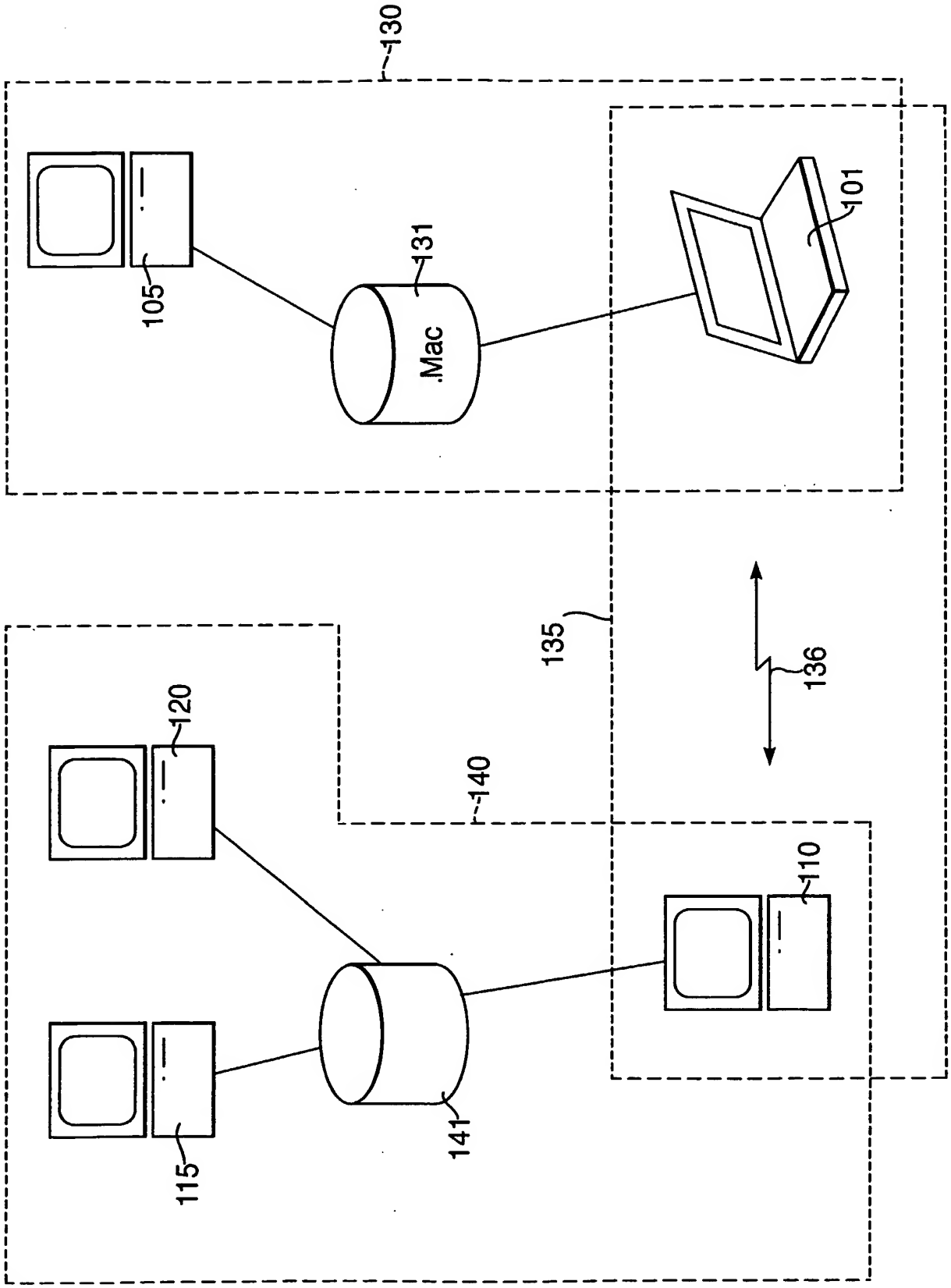


Fig.3A.

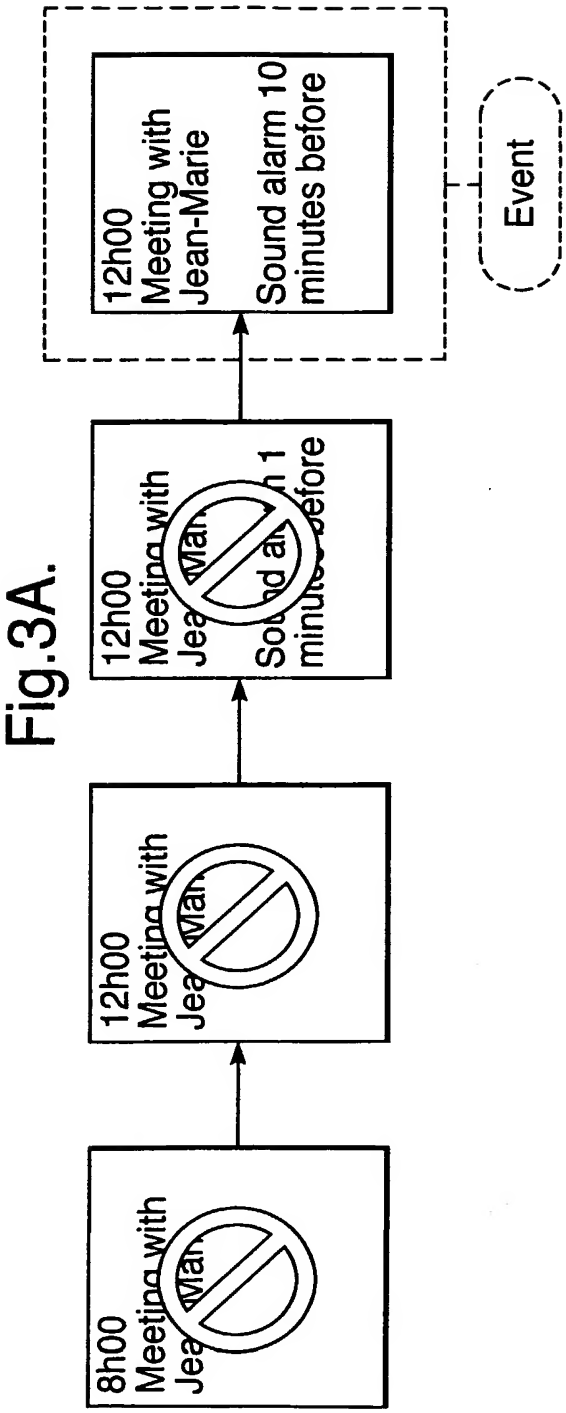

8h00
Meeting with
Jean-Marie
12h00
Meeting with
Jean-Marie
12h00
Meeting with
Jean-Marie
Sound alarm 1
minutes before
12h00
Meeting with
Jean-Marie
Sound alarm 10
minutes before
Event

Fig.3B.

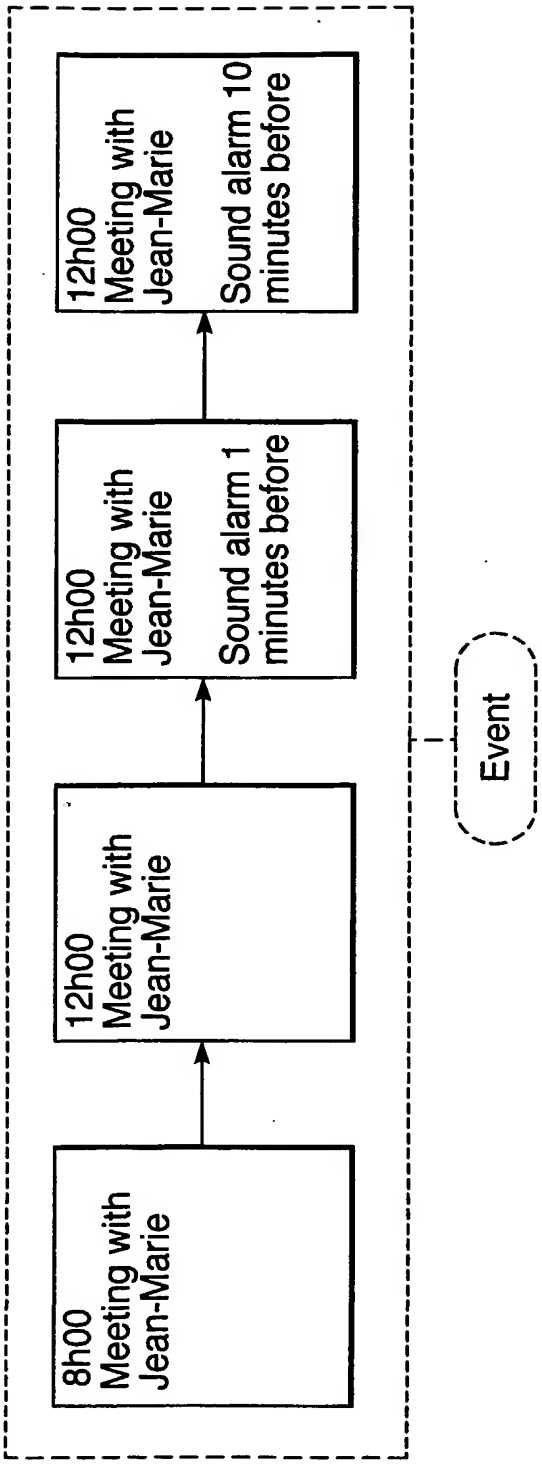

8h00
Meeting with
Jean-Marie
12h00
Meeting with
Jean-Marie
12h00
Meeting with
Jean-Marie
Sound alarm 1
minutes before
12h00
Meeting with
Jean-Marie
Sound alarm 10
minutes before
Event

Fig.4.

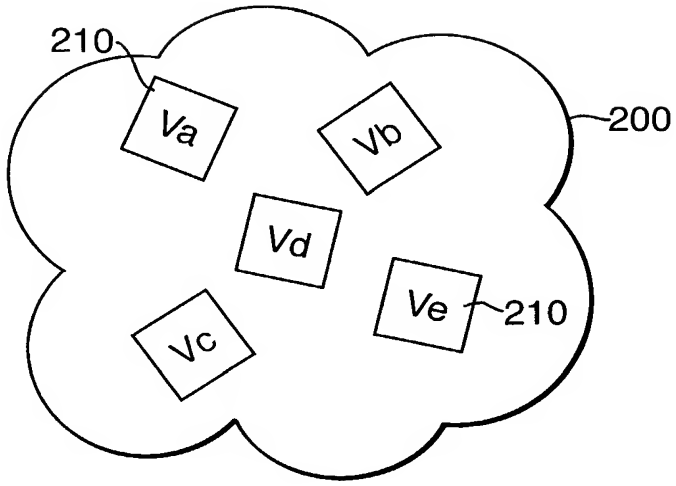


Fig.5A.

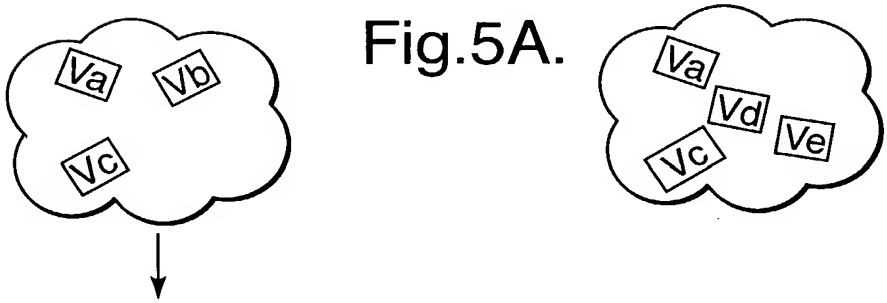


Fig.5B.

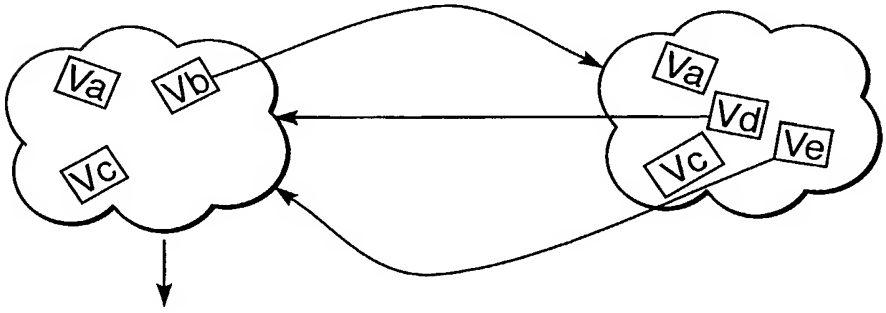


Fig.5C.

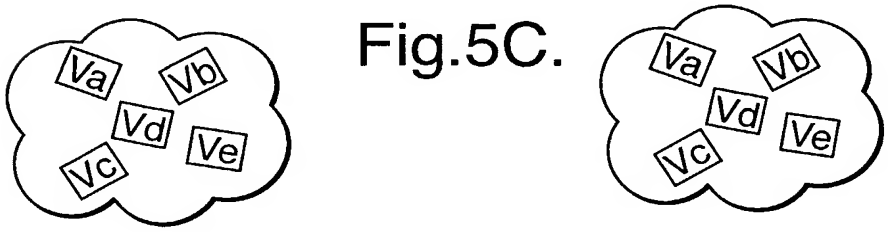


Fig.6.

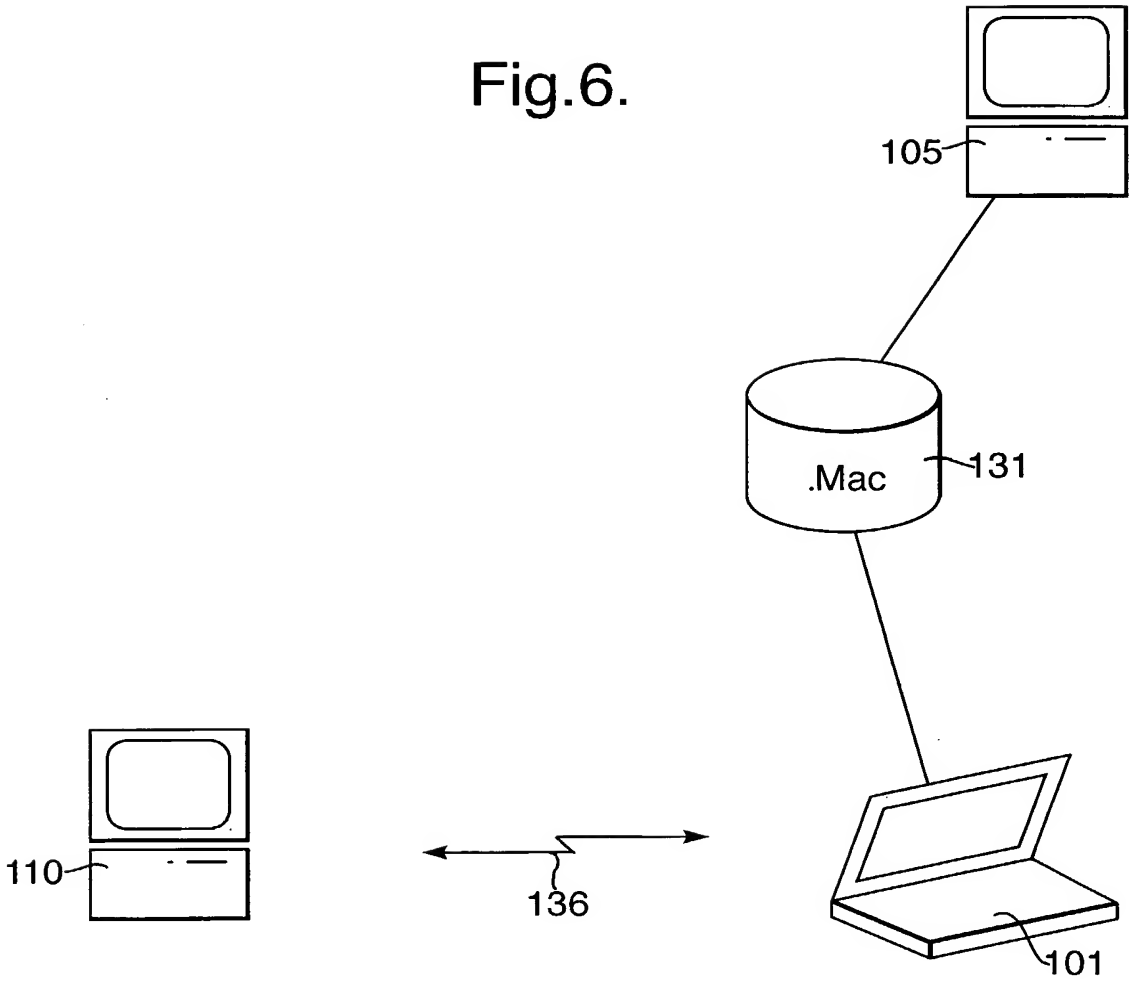


Fig.7.

Step	User 1	User 2	User 3
1	V_0	-	-
2	V_0	V_0	V_0
3	V_0	V_0 V_1	V_0
4	V_0 V_1	V_0 V_1	V_0
5	V_0 V_1	V_0 V_1	V_0 $V_{1'}$
6	V_0 V_1 $V_{1'}$	V_0 V_1 $V_{1'}$	V_0 $V_{1'}$ V_1
7	V_0 V_1 $V_{1'}$	V_0 V_1 $V_{1'}$	V_0 V_1 $V_{1'}$

Fig.8.

Step	User 1	User 2	User 3
1	W_0	-	-
2	W_0	W_0	W_0
3	W_0	W_0 W_1	W_0
4	W_0	W_0 W_1	W_0 $W_{1'}$
5	W_0 $W_{1'}$	W_0 W_1	W_0 $W_{1'}$
6	W_0 $W_{1'}$ W_1	W_0 W_1 $W_{1'}$	W_0 $W_{1'}$ W_1
7	W_0 W_1 $W_{1'}$	W_0 W_1 $W_{1'}$	W_0 W_1 $W_{1'}$

Fig.9.

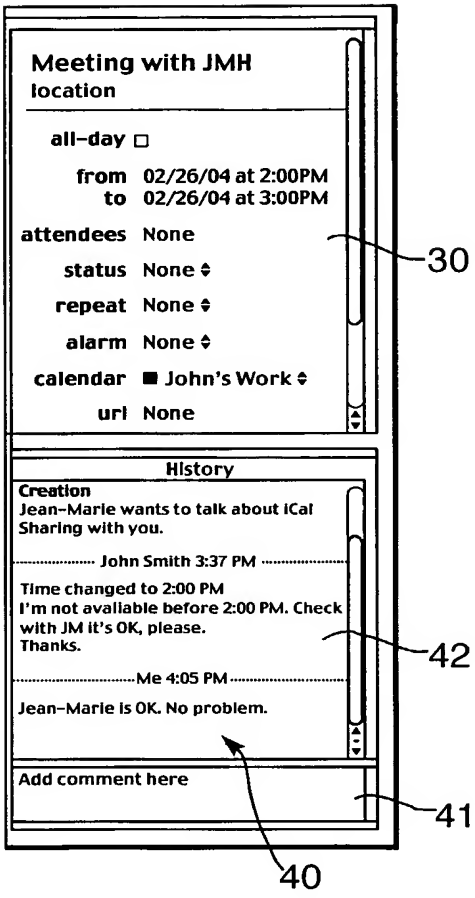

Meeting with JMH
location
all-day
from 02/26/04 at 2:00PM
to 02/26/04 at 3:00PM
attendees None
status None
repeat None
alarm None
calendar John's Work
url None
30
History
Creation
Jean-Marie wants to talk about iCal Sharing with you.
John Smith 3:37 PM
Time changed to 2:00 PM
I'm not avaliable before 2:00 PM. Check with JM it's OK, please.
Thanks.
Me 4:05 PM
Jean-Marie is OK. No problem.
42
Add comment here
41
40

Fig.10A.

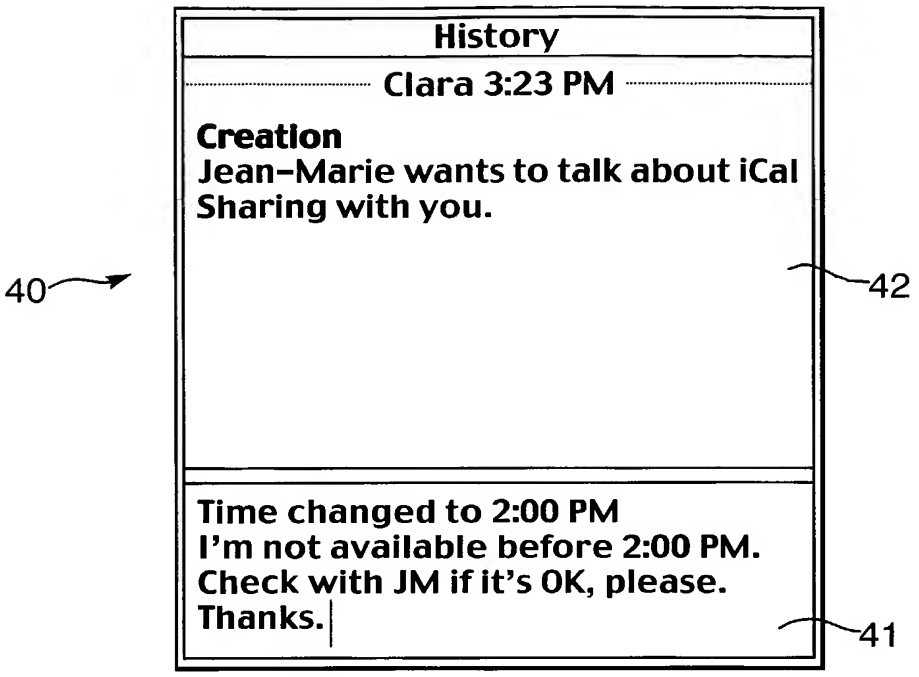


Fig.10B.

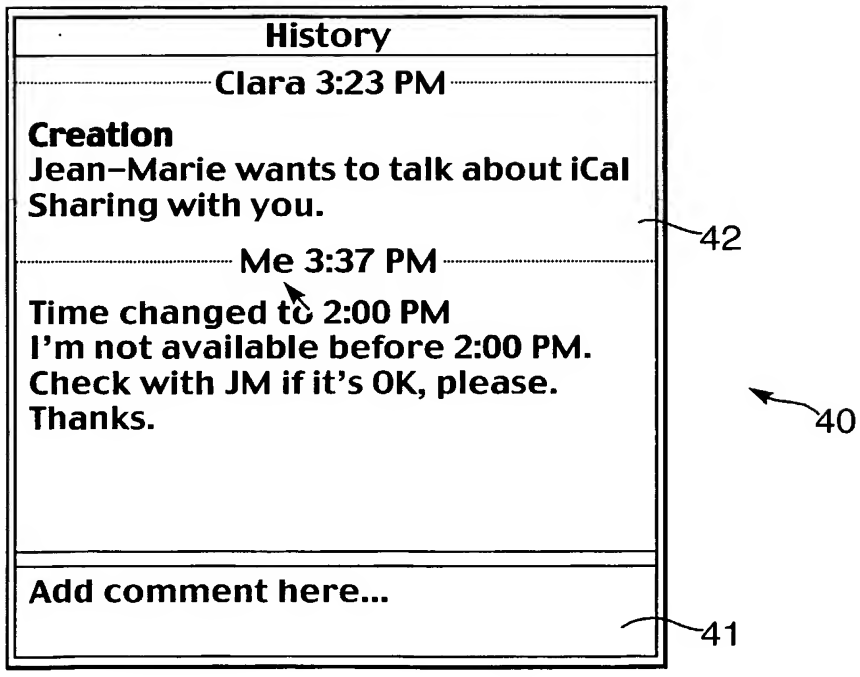


Fig.11.

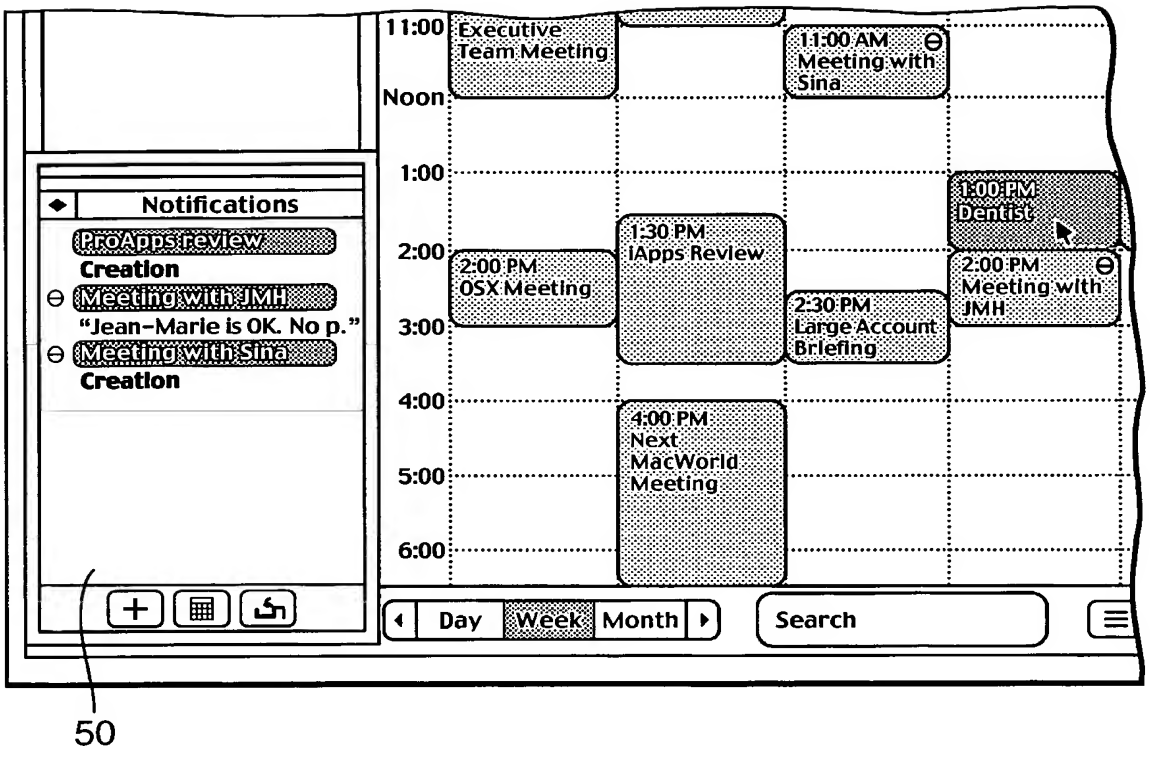
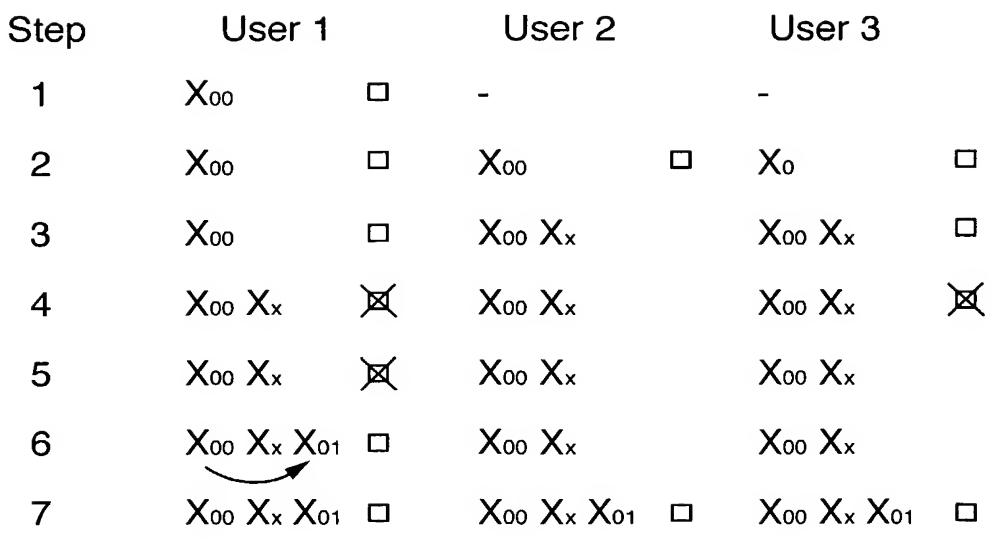

11:00
Noon
1:00
2:00
3:00
4:00
5:00
6:00
Executive Team Meeting
11:00 AM Meeting with Sina
1:00 PM Dentist
1:30 PM iApps Review
2:00 PM OSX Meeting
2:30 PM Large Account Briefing
2:00 PM Meeting with JMH
4:00 PM Next MacWorld Meeting
Notifications
ProApps review Creation
Meeting with JMH
"Jean-Marie is OK. No p."
Meeting with Sina
Creation
Day
Week
Month
Search
50

Fig.12.


Step
User 1
User 2
User 3
1
X_{00}
-
-
2
X_{00}
X_{00}
X_0
3
X_{00}
X_{00} X_x
X_{00} X_x
4
X_{00} X_x
X_{00} X_x
X_{00} X_x
5
X_{00} X_x
X_{00} X_x
X_{00} X_x
6
X_{00} X_x X_{01}
X_{00} X_x
X_{00} X_x
7
X_{00} X_x X_{01}
X_{00} X_x X_{01}
X_{00} X_x X_{01}

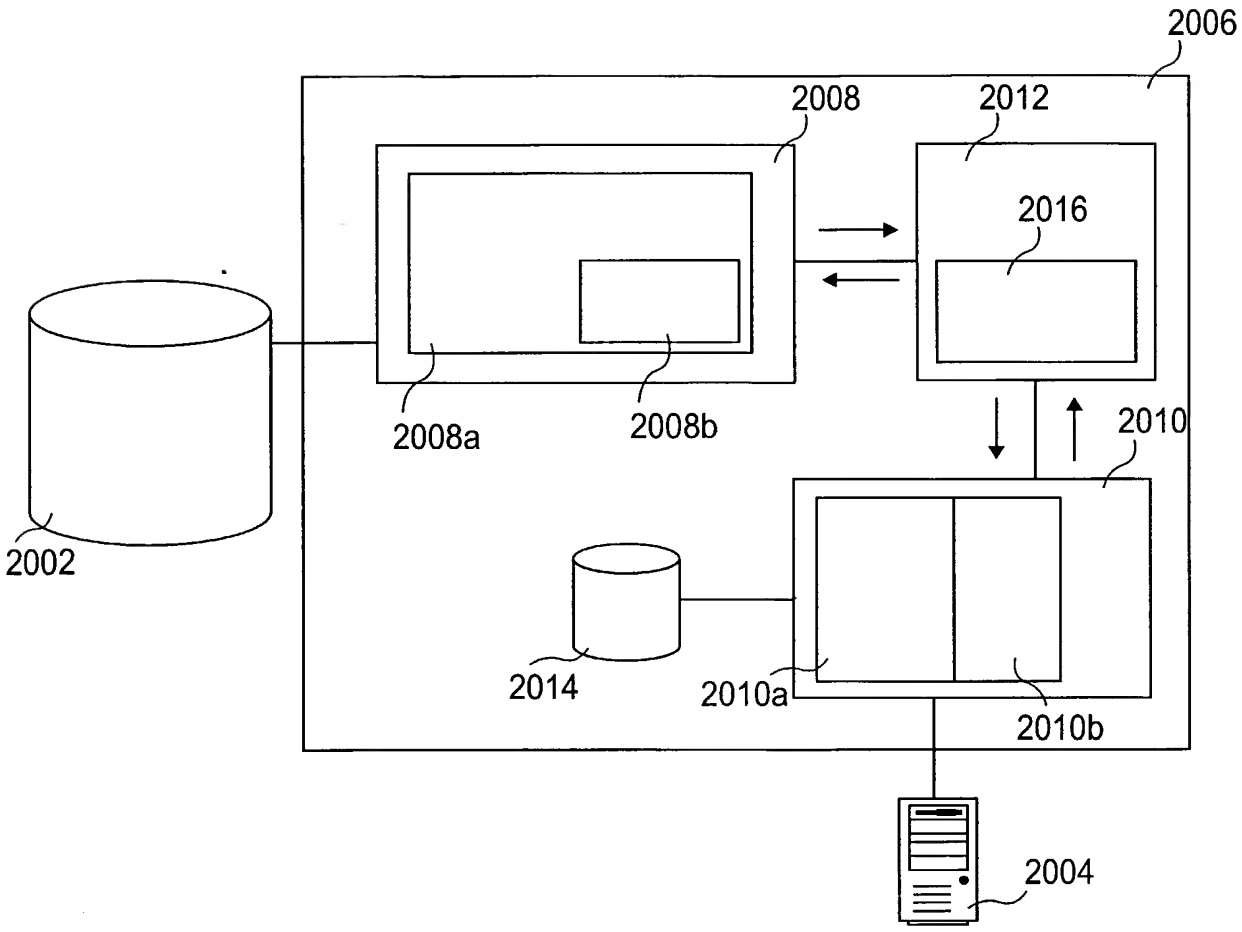

2006
2008
2012
2016
2008a
2008b
2010
2002
2014
2010a
2010b
2004

FIG. 13

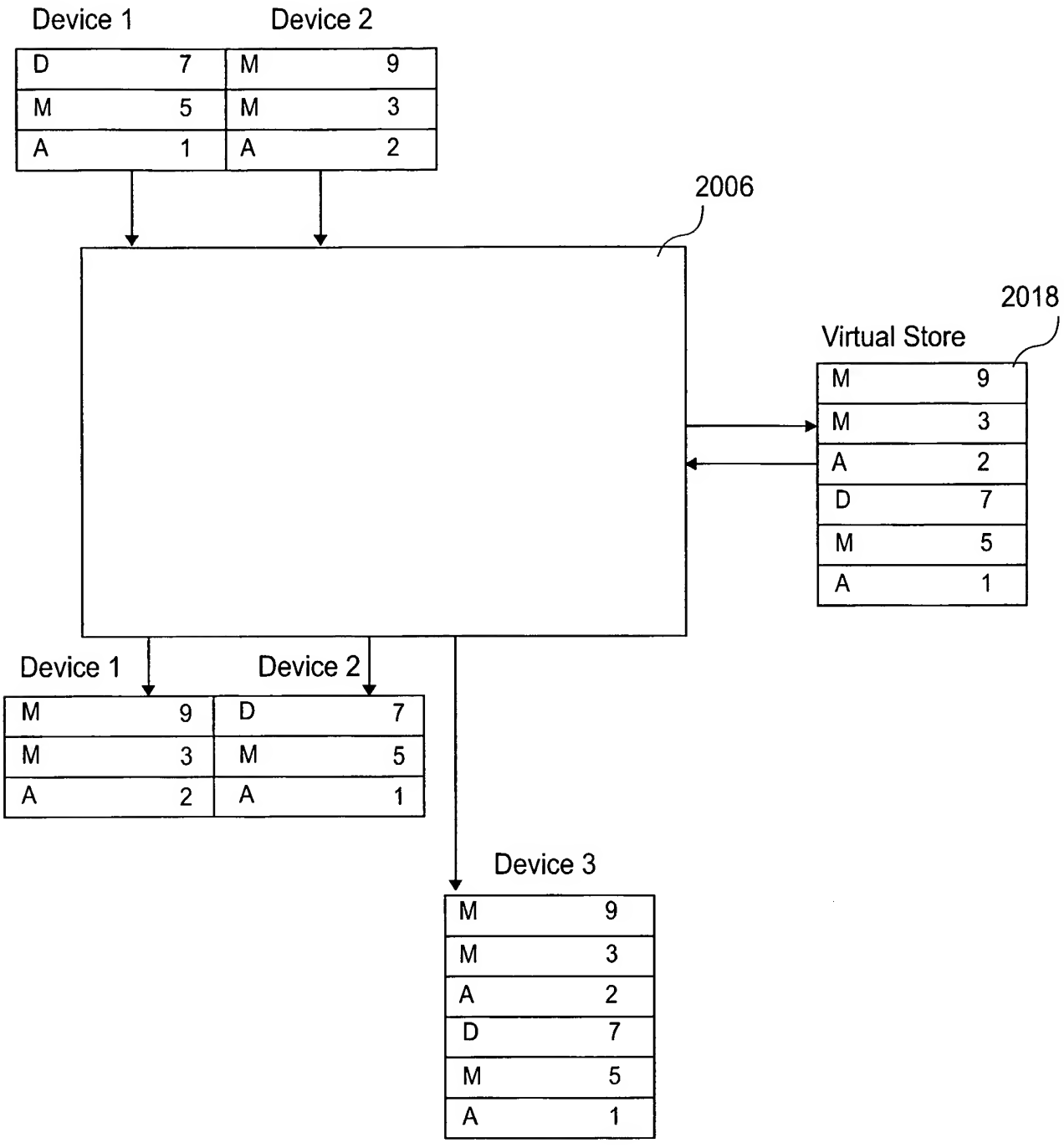

Device 1
Device 2
D 7
M 5
A 1
M 9
M 3
A 2
2006
2018
Virtual Store
M 9
M 3
A 2
D 7
M 5
A 1
Device 1
Device 2
M 9
M 3
A 2
D 7
M 5
A 1
Device 3
M 9
M 3
A 2
D 7
M 5
A 1

FIG. 14

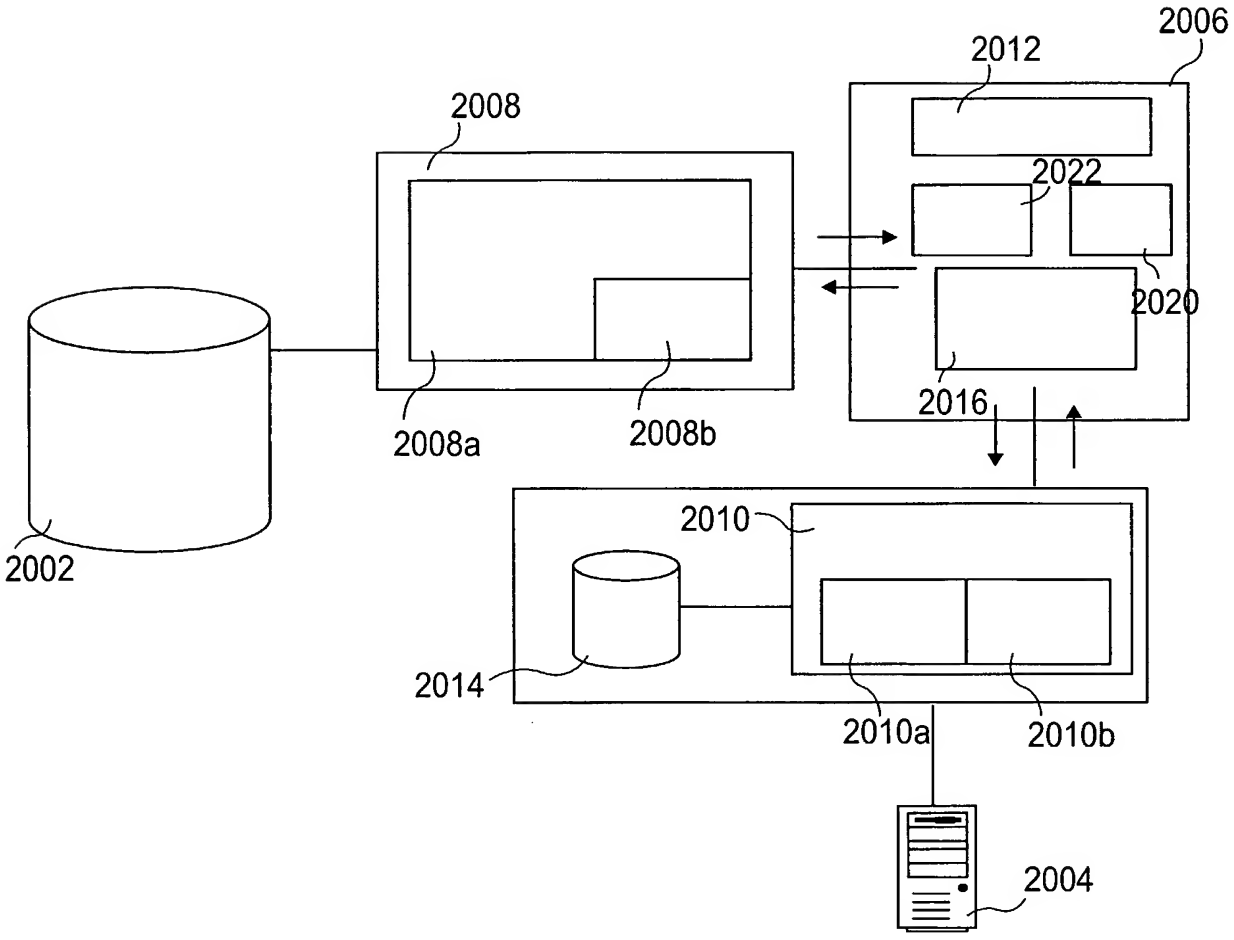

2006
2012
2008
2022
2020
2016
2008a
2008b
2002
2010
2014
2010a
2010b
2004

FIG. 15

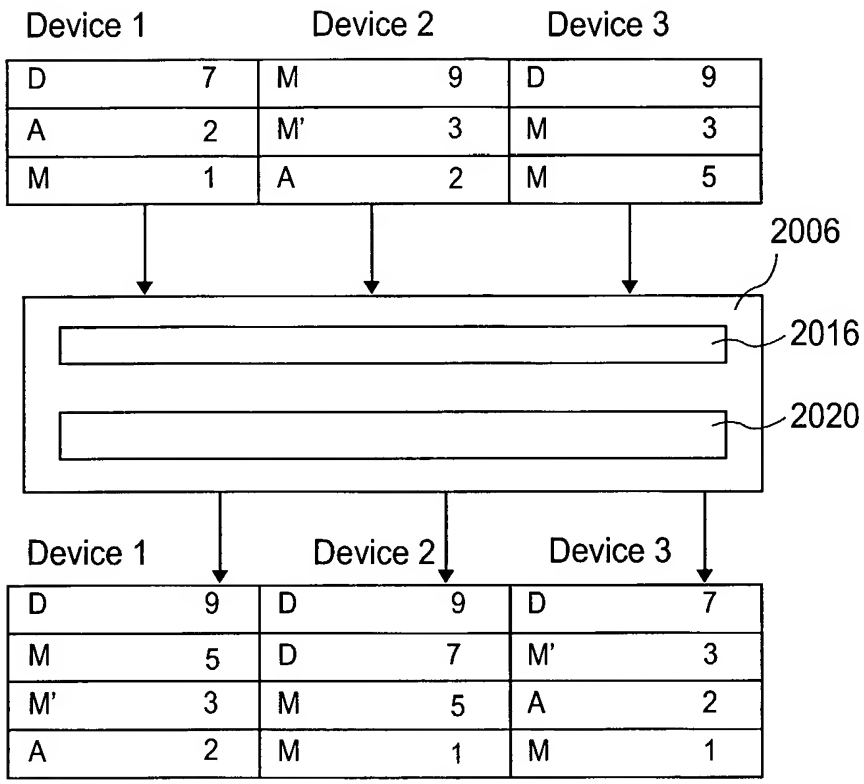

Device 1
Device 2
Device 3
D 7
M 9
D 9
A 2
M' 3
M 3
M 1
A 2
M 5
2006
2016
2020
Device 1
Device 2
Device 3
D 9
D 9
D 7
M 5
D 7
M' 3
M' 3
M 5
A 2
A 2
M 1
M 1

FIG. 16

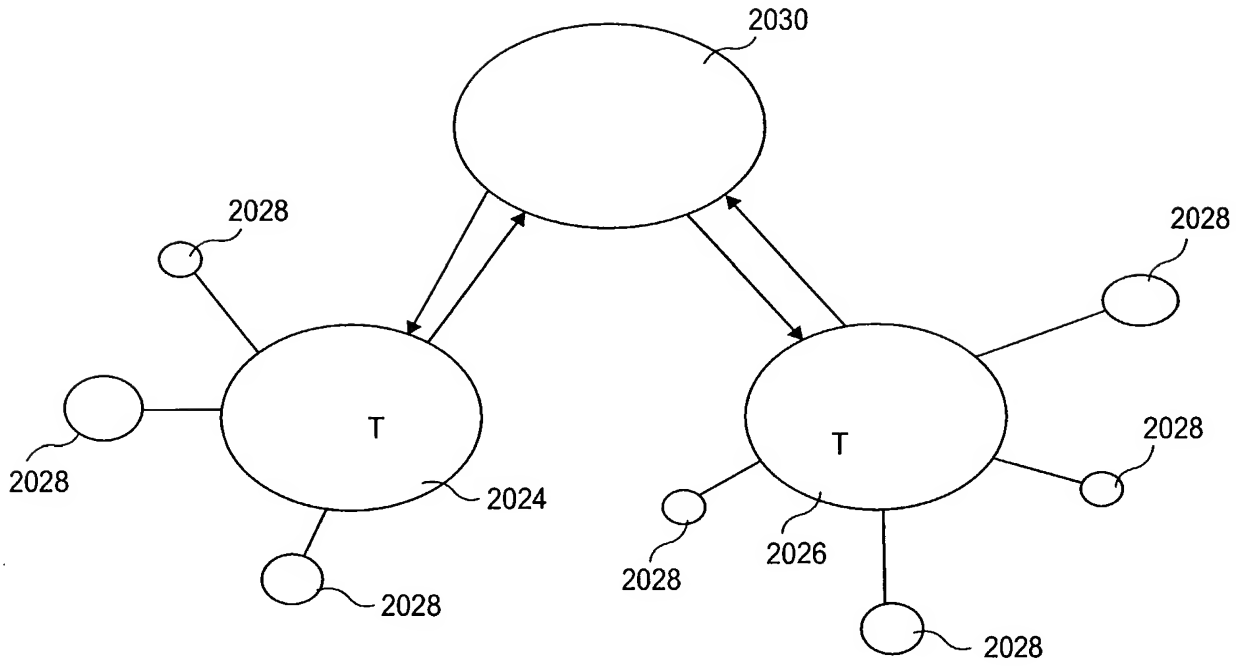


FIG. 17

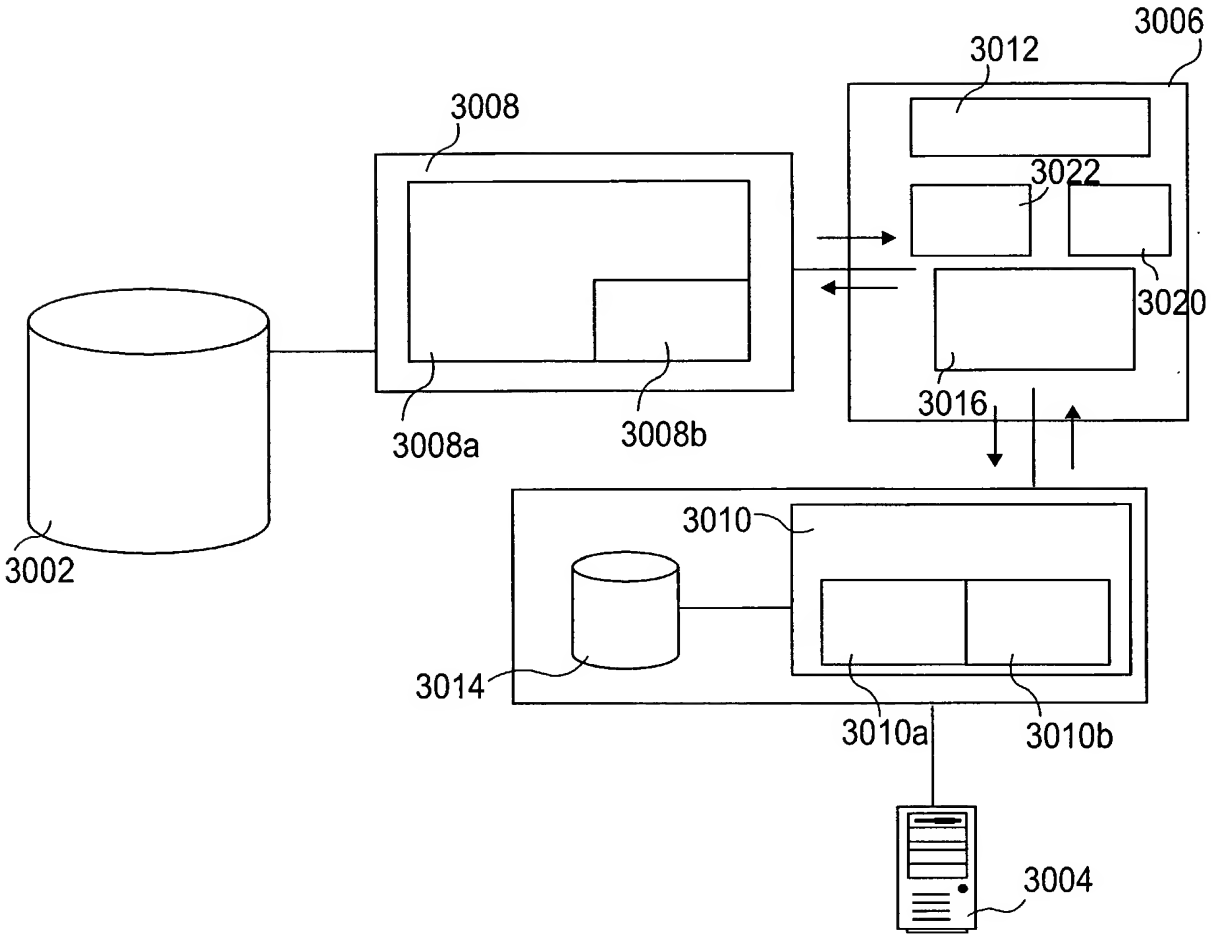

3006
3012
3008
3022
3020
3016
3008a
3008b
3002
3010
3014
3010a
3010b
3004

FIG. 18

Administration table

Client	Generation
Phone	0
Palm device	0
Home PC	0
Work PC	0

Truth table

Datum	Modify Generation	Add Generation	Client

FIG. 19

Administration table

Client	Generation
Phone	1
Palm device	0
Home PC	0
Work PC	0

Truth table

Datum	Modify Generation	Add Generation	Client
X		1	Phone
Y		1	Phone

FIG. 20

Administration table

Client	Generation
Phone	2
Palm device	0
Home PC	0
Work PC	0

Truth table

Datum	Modify Generation	Add Generation	Client
X		1	Phone
Y		1	Phone
Z		2	Phone

FIG. 21

Administration table

Client	Generation
Phone	3
Palm device	3
Home PC	3
Work PC	3

Truth table

Datum	Modify Generation	Add Generation	Client
X		1	Phone
Y		1	Phone
Z		2	Phone

FIG. 22

Administration table

Client	Generation
Phone	3
Palm device	3
Home PC	3
Work PC	3

Truth table

Datum	Modify Generation	Add Generation	Client

FIG. 23

Administration table

Client	Generation
Phone	4
Palm device	3
Home PC	3
Work PC	3

Truth table

Datum	Modify Generation	Add Generation	Client
A		4	Phone

FIG. 24

Administration table

Client	Generation
Phone	4
Palm device	5
Home PC	3
Work PC	3

Truth table

Datum	Modify Generation	Add Generation	Client
A		4	Phone
B		5	Palm

FIG. 25

Administration table

Client	Generation
Phone	6
Palm device	5
Home PC	3
Work PC	3

Truth table

Datum	Modify Generation	Add Generation	Client
A		4	Phone
B		5	Palm
C		6	Phone

FIG. 26

Administration table

Client	Generation
Phone	7
Palm device	7
Home PC	7
Work PC	3

Truth table

Datum	Modify Generation	Add Generation	Client
A		4	Phone
B		5	Palm
C		6	Phone
D		7	Home PC

FIG. 27

Administration table

Client	Generation
Phone	8
Palm device	8
Home PC	8
Work PC	8

Truth table

Datum	Modify Generation	Add Generation	Client

FIG. 28

Administration table

Client	Generation
Phone	9
Palm device	8
Home PC	8
Work PC	8

Truth table

Datum	Modify Generation	Add Generation	Client
G'	9		Phone

FIG. 29

Administration table

Client	Generation
Phone	9
Palm device	8
Home PC	8
Work PC	10

Truth table

Datum	Modify Generation	Add Generation	Client
G'	9		Phone
G''	10		Work PC

FIG. 30

Truth attribute table

Record No.	Datum	Modify Generation	Add Generation	Client
1	S''	11		Work PC
1	Q	11		Work PC
2	T	12		Home PC

Truth relationship table

Relationship	Source from	Target to	Modify Generation	Add Generation	Client
to parent	1	2			
to child	2	1			
to child	2	3			

FIG. 31

INTERNATIONAL SEARCH REPORT

Interr ıal Application No
PCT/US2005/014619

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F17/60

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 02/089026 A (HEWLETT-PACKARD COMPANY; SELLEN, ABIGAIL, JANE; MORGAN, ANDREW, DUDLEY) 7 November 2002 (2002-11-07) abstract page 1, line 5 - page 6, line 23 page 8, line 29 - page 11, line 6 page 12, line 16 - page 14, line 26 page 15, line 21 - page 17, line 4 page 18, lines 8-22 page 20, line 10 - page 22, line 5 figures 8,9 ----- -/--	1-59

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
29 August 2005	02/09/2005

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Peelen, B

Form PCT/ISA/210 (second sheet) (January 2004)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	PRASAD S K ET AL: "Implementation of a calendar application based on SyD coordination links" PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM, 2003. PROCEEDINGS. INTERNATIONAL APRIL 22-26, 2003, PISCATAWAY, NJ, USA,IEEE, 22 April 2003 (2003-04-22), pages 242-249, XP010645361 ISBN: 0-7695-1926-1 abstract paragraph '0001! paragraphs '03.2! - '04.2! paragraphs '0005! - '0007!	1-59
X	PRASAD S K ET AL: "Enforcing interdependencies and executing transactions atomically over autonomous mobile data stores using SyD link technology" MULTIMEDIA SIGNAL PROCESSING, 2002 IEEE WORKSHOP ON 9-11 DEC. 2002, PISCATAWAY, NJ, USA,IEEE, 19 May 2003 (2003-05-19), pages 803-809, XP010642468 ISBN: 0-7803-7713-3 abstract paragraph '02.1! paragraphs '0005!, '0006!	1-59
X	US 2003/045301 A1 (WOLLRAB LEE M) 6 March 2003 (2003-03-06) abstract paragraph '0001! paragraphs '0008! - '0011! paragraphs '0020! - '0026! paragraphs '0030!, '0034!, '0035! claims 1,4-6,10	1-59
X	WO 02/44958 A (TELEFONAKTIEBOLAGET L M ERICSSON ; NOVAK, LARS; BIRKLER, JOERGEN) 6 June 2002 (2002-06-06) abstract page 1, line 6 - page 3, line 2 page 6, line 27 - page 7, line 16 figure 6	1-59

-/--

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US2005/014619

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	BISIGNANO M ET AL: "Expeerience: a JXTA middleware for mobile ad-hoc networks" PEER-TO-PEER COMPUTING, 2003. (P2P 2003). PROCEEDINGS. THIRD INTERNATIONAL CONFERENCE ON 1-3 SEPT 2003, PISCATAWAY, NJ, USA,IEEE, 1 September 2003 (2003-09-01), pages 214-215, XP010657550 ISBN: 0-7695-2023-5 the whole document	1-59
A	PALUSKA J M ET AL: "Footloose: a case for physical eventual consistency and selective conflict resolution" MOBILE COMPUTING SYSTEMS AND APPLICATIONS, 2003. PROCEEDINGS. FIFTH IEEE WORKSHOP ON 9-10 OCT. 2003, PISCATAWAY, NJ, USA,IEEE, 9 October 2003 (2003-10-09), pages 170-179, XP010662885 ISBN: 0-7695-1995-4 the whole document	1-59

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US2005/014619

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 02089026 A	07-11-2002	GB 2375017 A	30-10-2002
		EP 1393214 A2	03-03-2004
		WO 02089026 A2	07-11-2002
		US 2004093380 A1	13-05-2004
US 2003045301 A1	06-03-2003	NONE	
WO 0244958 A	06-06-2002	US 2002069298 A1	06-06-2002
		AT 298112 T	15-07-2005
		AU 3722302 A	11-06-2002
		DE 60111562 D1	21-07-2005
		WO 0244958 A1	06-06-2002
		EP 1340176 A1	03-09-2003